

المؤسسة العامة للكتاب والنشر والتوزيع



الدكتور عبد الحسن الحسيني

المساعد في البرمجة « بازيك »

0009016



Bibliotheca Alexandrina

المساعد في البرمجة
« بازيك »

جميع الحقوق محفوظة

الطبعة الثانية

1407 هـ - 1987 م

المؤسسة الجامعية للدراسات والنشر والتوزيع

بيروت - الحمراء - شارع اميل الهدهد - بناية سلام

هاتف ٨٠٢٤٢٨ - ٨٠٢٤٠٧ - ٨٠٢٤٩٦

بيروت - المصيطة - بناية طاهر - هاتف : ٣٠١٠٣٠ - ٣١١٣١٠

ص.ب. ١١٣ / ٦٣١١ - ٢٠٦٦٥ L.E - ٢٠٦٨٠ - لسان

سلسلة بإشراف
د. عبد الحسن الحسيني

المساعد في البرمجة « بازيك »

الدكتور عبد الله بن الحسيني

طبعة ثانية منقحة ومزودة

المؤسسة الجامعية للدراسات والنشر والتوزيع

المساعد في البرمجة « بازيك »

PROGRAMMING GUIDE

— BASIC —

هذا الكتاب يعطي صورة واضحة عن طريقة البرمجة بلغة « بازيك » (BASIC) . وهو مفيد لطلاب علوم الكمبيوتر والبرمجة ، وبالأخص لكل من يرغب في تعلم البرمجة بهذه اللغة والعمل بها . ويحتوي هذا الكتاب على ثلاثة فصول :

الفصل الأول : عبارة عن مدخل إلى البرمجة ، ويعطي صورة موجزة عن البرمجة وكيفية وضع البرامج .

الفصل الثاني : يُعتبر مدخل إلى لغة « بازيك » ، ويعطي صورة واضحة عن هذه اللغة مع الكثير من الأمثلة التي تساعد القارئ على فهم هذه اللغة والعمل بها .

الفصل الثالث : ويحتوي على الأوامر والتعليمات المستعملة لتنظيم وإدارة السجلات والعمل بالأجهزة .

الفصل الرابع : ويحتوي على عدد كبير من المسائل المُحلَّلة والمبرمجة ، التي تساعد القارئ في عمله بهذه اللغة .

ولقد حاولت قدر الإمكان المحافظة على المصطلحات الإنكليزية المعتمدة في الكمبيوتر ، بالإضافة إلى المصطلحات العربية الجديدة التي دخلت إلى الكمبيوتر منذ مدة قريبة .

ومنه سبحانه نستمد العون والتوفيق

الفصل الأول

البرمجة

PROGRAMMING

البرمجة

الآلة الحاسبة، بشكل عام هي عبارة عن صناديق صامتة معبأة بالإلكترونيات ، إذا لم تتم برمجتها . ولكي نستطيع استعمال هذه الآلات بالشكل الصحيح والاستفادة من مقدراتها ، يجب أن تحتوي على نظامٍ للتشغيل (Operation System) ، وعلى رزمة من البرامج الرياضية والفيزيائية والعلمية . . . تساعد المبرمج في وضع برامج وتسهّل عليه عمله (Software) ، عندها تصبح الآلة جاهزة للعمل والقيام بالمهام المطلوبة منها ، ولكن هذه المهام تُلقن للآلة بواسطة برامج مكتوبة بلغات خاصة تفهمها وتنفذها بواسطة نظام التشغيل الذي تعمل به .

وهكذا فالبرمجة هي أساس استعمال الآلة ، وعندما يكون البرنامج صحيحاً تقوم الآلة بأداء مهامها على أكمل وجه . وليس من الضروري أن يعبر البرنامج عن معادلة رياضية صعبة أو مُعقّدة، ولكن يجب أن يكون عبارة عن تسلسل منطقي لعدد من المعادلات والدالات الرياضية . بالإضافة لعدد من أوامر تنظيم عملية الحساب داخل الآلة . وعندما يفكر المبرمج أو المهندس بوضع برنامج مُعين لمسألته يجب أن يدرك الأمور التالية :

- أ - أن يفهم المسألة فهماً صحيحاً ، وأن يحدد ما يريد منها وما يريد من النتائج .
- ب - أن يُحدّد المعطيات (Data) المطلوب تلقيها للبرنامج وللآلة ، ووضع بنك المعلومات (Data Base) المطلوب لعملية تنفيذ البرنامج والحصول على النتائج .
- ج - أن يتأكد من إمكانية وضع برنامج للمسألة ، ومن مقدرة اللغة التي اختارها للتعبير عن حلّ المسألة أو عن خوارزم حلّ المسألة .
- د - أن يُحدّد العمليات المطلوب إجراؤها .
- هـ - أن يتأكد من إمكانية الآلة الحاسبة على إستيعاب البرنامج وحله . فالآلات تختلف بمقدراتها وبحجم ذاكرتها الداخلية وسرعة تحليل المعلومات . فالآلة

الكبيرة مثلاً ، تتمتع بذاكرة داخلية كبيرة وبسرعة أكبر في تحليل البرنامج وحله ، أما الميني والميكرو كومبيوتر فذاكرتهم الداخلية أقل حجماً . وتنفيذ البرامج يتطلب وقتاً أطول .

و- أن يتطلع على الوثائق التابعة للآلة والموجودة عنده ، وعلى نظام التشغيل (Operation System) وعلى الكتب المساعدة للمبرمج في عمله وعلى رزمة البرامج الموجودة في مكتبة البرامج (Software) والتي من الممكن الاستفادة منها عند كتابة برنامجه . ذلك إن الشركات المنتجة تتطور عادة اللغة المستعملة . فتزيد من الأوامر والتعليمات (Instruction) أو تنقّصها ، حسب ما تعتبره هي مفيداً ، آخذة بعين الاعتبار مقدرة الآلة المنتجة وخصائصها .

ومن المعلوم أن أنظمة التشغيل هي متعددة (DOS, TOS, OS, VS...) وأوسعها إنتشاراً نظامي DOS و OS. هذه الأنظمة تدير عمليات الآلة ، وتجعلها جاهزة للاستعمال وتعطي الاشارات حول الأخطاء النحوية الواقعة ضمن البرنامج (Syntax error) . وبواسطة نظام التشغيل أيضاً ، يستطيع المبرمج من أن يستعمل بعض التعليمات الكبرى الخاصة بالآلة Macro Instruction ، الموضوع بلغة المؤول الكُبرى (Macro Assembler) لتنظيم وإدارة الآلة حسب خطة معينة للتشغيل والصيانة ولزيادة مقدرة اللغة المختارة والمستعملة في برمجة المسألة . كما ويحتوي نظام التشغيل على برامج خاصة للتنقيح (Editor, Redactor, Translator) والترجمة ، من اللغة المستعملة إلى لغة الآلة . . وغير ذلك . لذلك يُعتبر نظام التشغيل من الأقسام المهمة في عمل الآلة ويباع معها كجزء لا يتجزأ منها وبدونه لا تستطيع الآلة القيام بأي عمل . وهكذا وللقيام بوضع برنامج معين لحل مسألة معينة ، يجب على المهندس أو المبرمج القيام بالعمليات التالية :

- 1 - تحليل المسألة Problem Analysis .
- 2 - وضع الخوارزم الخاص بالمسألة Algorithm, flow sharting .
- 3 - تكويد البرنامج Coding بلغة معينة ، أي ترجمة الخوارزم إلى سلسلة تعليمات رمزية مكتوب بإحدى لغات البرمجة .
- 4 - ترجمة البرنامج من اللغة الرمزية أو اللغة المتطورة إلى لغة الآلة .

- 5 - تصحيح واختيار البرنامج Testing .
6 - توثيق البرنامج Documentation وتسجيل الملاحظات الخاصة بكيفية استعمال البرنامج كعمل تطبيقي .
والآن نرى بإيجاز ما تعني كل نقطة من هذه النقاط .

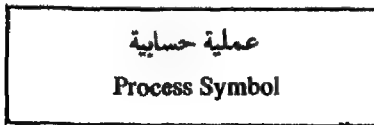
1 - تحليل المسألة Analysis .

- قبل البدء بعملية البرمجة ، يجب على المبرمج أو المهندس أن يُجيب على عدد من الأسئلة أهمها :
- أ - ماذا تفعل هذه المسألة .
ب - ما هو الغرض من حلها .
ج - ما هي النتائج المطلوب الحصول عليها .
د - هل نستطيع حل هذه المسألة بواسطة الكمبيوتر .
و - هل يقدر الكمبيوتر الموجود معنا ، على استيعاب البرنامج الموضوع لحل هذه المسألة .
ز - ما هي المعطيات (Data) المطلوب تلقيها للبرنامج ، وشكل المعطيات الداخلة ، وما هي المعلومات Out put information المطلوب إخراجها ، وكيف سيتم إخراجها وعلى ماذا .
ح - ما هي اللغة الأفضل في كتابة البرنامج ، أي اللغة التي تستطيع إحتواء جميع أقسام الخوارزم والتعبير عنه بشكل مفهوم صحيح واضح ومحدد .
ت - هل نستطيع برمجة جميع أجزاء الخوارزم أو المسألة باللغة المختارة ، وكيف بإمكاننا تجزئة المسألة إلى أقسام عديدة تساعدنا في كتابة البرنامج .
س - المعادلات الرياضية الأساسية التي سنستعملها في حل المسألة .

هذه الأمور يجب أن نحاول الإجابة عليها قبل البدء بعملية البرمجة ، وهي تساعد في وضع التسلسل المنطقي « الخوارزم » للبرنامج . كما ويجب وضع تصميم خاص بأشكال المعلومات الداخلة والخارجة ، وأقسام كل وحدة معلوماتية وتنظيم عملية إدخالها وإخراجها ، وتحديد الجهاز المستعمل في الإدخال والإخراج .

2 - التسلسل المنطقي للبرنامج أو الخوارزم . Algorithm

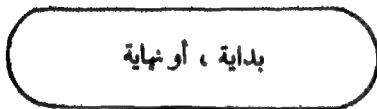
يعتبر الخوارزم من المراحل المهمة في البرمجة ، فهو يسهل من عملية كتابة البرامج باللغة المختارة ، ويخفف من عدد الأخطاء اللغوية والحسابية والمنطقية الممكن الوقوع بها . ويأتي وضع الخوارزم بعد مرحلة التحليل وإدراك وفهم المطلوب الحصول عليه من المسألة والبرنامج . وبواسطة الخوارزم يقوم المبرمج أو المحلل بتجزئة المسألة إلى عدة مسائل ، والعمليات إلى عدة عمليات أقل حجماً وصعوبة ، بحيث تصبح المسألة عبارة عن عدد من الخطوات الحسابية والعملية والرياضية والمنطقية ، تسير بشكل منطقي باتجاه حل المسألة . ولوضع الخوارزم نستعمل الرموز والرسوم التالية :



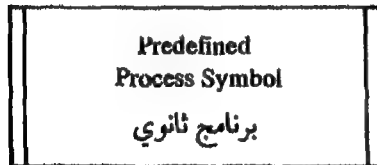
أ - يستعمل لكتابة ، أو للتصريح عن العملية أو مجموعة العمليات الحسابية ، أو الرياضية أو المنطقية المطلوب إجراؤها . كما وعن عمليات التخزين بالذاكرة والإخراج منها .



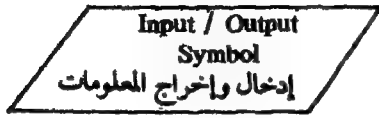
ب - يستعمل لاتخاذ قرار ما ، نتيجة تنفيذ شرط معين أو عدم حدوثه . وبواسطته نستطيع تغيير مسار البرنامج المنطقي وإحداث عمليات إضافية ناتجة عن تنفيذ الشرط أو عدم حدوثه .



ج - يستعمل لتحديد بداية أو نهاية البرنامج .



د - يستعمل للتصريح عن ، أو دعوة برنامج ثانوي معين . أو إدخال أقسام وبرامج جاهزة في البرنامج الرئيسي .



هـ - يستعمل للتصريح عن عمليات الإدخال والإخراج للمعلومات .

Connector Symbol



و - عبارة عن وصلة بين أجزاء الخوارزم .

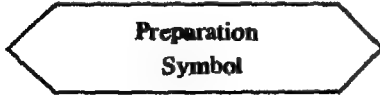
Flow line



ز - يستعمل للإتصال بين أجزاء الخوارزم .



ح - يستعمل لإضافة شروحات حول البرنامج .



ت - يستعمل لتحضير خلايا معينة للعمل ولإعطاء قيمة بدائية للمتحولات .

3 - تكويد البرنامج أو كتابة البرامج بلغة معينة (Coding)

تتألف كل لغة من اللغات من كلمات ، وجمل ، وأرقام .

وهناك اللغات الرمزية (Symbolic) كلغة المؤول (Assembler) ولغة الآلة (Machine language) . وهذه اللغات هي غير مفهومة للقارئ وعبارة عن رموز معينة تشير إلى عمل معين تفهمه الآلة وتقوم بتنفيذه . وهناك أيضاً اللغات القريبة من اللغة المحكية ، وتسمى « لغات ذات مستوى مرتفع أو لغات متطورة » (High level language) . كلغة « بازيك » ، باسكال ، PL/ 1 ، GPSS ، الخ . وهكذا فبعد وضع خوارزم المسألة ، يتم اختيار اللغة المناسبة حسب رأي المبرمج ، والبدء بكتابة البرنامج بحيث إن كل رسم من الخوارزم يؤلف واحداً أو عدداً من التعليمات (Instruction) . ويتم عادة ترقيم الأوامر وتلقينها للآلة بشكل مجمل ، كل منها يكتب على سطر واحد ويتم إدخالها إلى الآلة إما بواسطة مفاتيح الشاشة التلفزيونية (Terminal Key) ، وإما بواسطة البطاقات المثقوبة (Punch Card) أو بطريقة أخرى .

وعند تلقين أو إدخال البرنامج للآلة ، يمكن أن تحدث معنا الأخطاء

التالية :

- أخطاء نتيجة عملية تثقيب البطاقات أو في إدخال التعليمات بواسطة لوحة الملامس .
- نتيجة إرتباك في الرموز وإدخال واحدة بدلاً من أخرى مثلاً : ϕ و D .
- رموز غير موجودة في الكود المستعمل مثلاً EBCIDE ، أو غير مفهومة ومعروفة من قبل الآلة .
- قرارات غير موضوعية أو منطقية .
- سقوط أحد فروع البرنامج سهواً .
- 4 - الترجمة Translation والتصريف .

تتم عملية الترجمة بواسطة برنامج خاص يدعى مُصَرِّف (Compiler) موجود بداخل نظام التشغيل . وهذا البرنامج يقوم بترجمة البرنامج من اللغة المكتوب بها إلى « لغة الآلة » التي تفهمها الآلة . ولكل لغة من اللغات برنامج خاص بها للتصريف والترجمة . وأثناء عملية الترجمة يقوم نظام التشغيل بإكتشاف الأخطاء وإعطاء الإشارة إلى الأخطاء اللغوية (Syntax error) المرتكبة أثناء كتابة البرنامج . وبعد تصحيح الأخطاء نحصل على البرنامج الموضوعي المستهدف الذي تظمه الآلة ويُسمى Object program . وهو عبارة عن أرقام سادس عشرية أو ثنائية ، كل عدد يؤلف كلمة تشير إلى أمر معين أو عمل معين . بعد ذلك تجري معالجة هذا البرنامج بواسطة مُنقِّح الأربطة (LINK EDITOR) الذي يربط التعليمات فيما بينها وعند ذلك نحصل على البرنامج الموضوعي القابل للتنفيذ .

وأثناء تنفيذ البرنامج تقوم الآلة بالإشارة لعدد آخر من الأخطاء المنطقية (Logic and diagnostic error) ، ويجب على المبرمج تصحيحها وإعادتها إلى الآلة ، حتى يحصل على البرنامج الحقيقي الصحيح .

إضافة لهذه العملية المركبة ، فهناك إمكانية لمعالجة البرامج المكتوبة بلغة 'البازيك مباشرة بواسطة منهاج مسجل في الذاكرة الثابتة للمكنة ويدعى مُفسِّره (Interpreter) .

هذا البرنامج يقوم بتفسير تلقائي وأتوماتيكي لكل تعليمة وتحويلها إلى تعليمة المكنة في الوقت الذي يتم فيه إدخال هذه التعليمة إلى المكنة . إستعمال هذه

المفسرات لا يزال مقصوراً على لغة « البازيك » وفي بعض الأحيان لغة « الباسكال » وتحديدًا في الميكروكومبيوترات . وتتميز هذه البرامج ببطء تنفيذها نسبة إلى البرامج المُصرَّفة بواسطة المُصرِّف .

5 - تصحيح الأخطاء Testing

بعد ترجمة البرنامج إلى لغة الآلة ، يقوم المبرمج بتلقيم هذا البرنامج الموضوعي المستهدف القابل للتنفيذ (Object program) ، بالمعطيات اللازمة والتي تتألف من أعداد ورموز (Data information) ، وذلك للتأكد من عمل البرنامج ، ولرابعة النتائج والأخطاء المنطقية المرتكبة . وهكذا فإذا كانت النتائج صحيحة وأكيدة ، فمعناه أن البرنامج صحيحاً وسيكون جاهزاً للاستعمال حيث تدعو الحاجة لذلك ، ويمكن استعماله وتلقيه المعطيات الحقيقية للحصول على النتائج المطلوبة حيث تدعو الحاجة له . وفي الحالة الأخرى ، يقوم المبرمج بمحاولة تصحيح الأخطاء المنطقية (Logic error) ، والتي من المتوقع أن تكون السبب في عدم تنفيذ البرنامج وعدم صحته . وتتم عملية التصحيح بعد قراءة « كلمة حالة البرنامج » أو الرسالة الخاصة بالخطأ (Program Status Word = P S W) التي يقوم الحاسب بإخراجها على المطبعة أو على الشاشة التلفزيونية للميكروكومبيوتر أو الآلة الكاتبة للآلة الكبيرة . كما ومن المفروض قراءة وإخراج مخزون الذاكرة والمسجلات الداخلية للحاسب المركزي DUMP لمقارنة ومتابعة مضمونها في محاولة لفهم سبب الخطأ . وهذه العملية تدعى Testing and debugging .

6 - التوثيق Documentation .

بعد التأكد من صحة عمل البرنامج ، يتم وضع شروحات حول عمله ، والطريقة الرياضية المستعملة ، والتسلسل المنطقي له ، وطريقة استعماله واستدعائه للعمل ، وكيفية تلقيه بالملومات وتوضيح الفاييل والتسجيلات المستعملة (Description of files and records) .

يتم كتابة هذه الأشياء جميعها وحفظها مع لائحة الأوامر المطبوعة من قبل الآلة (Listing) . كما يتم تخزين البرنامج أو تسجيله على شريط مغناطيسي أو ديسك مغناطيسي لاستعماله عند الحاجة .

هذه العملية بأكملها تدعى Documentation .

الفصل الثاني

المدخل إلى البرمجة بلغة « بازيك »

INTRODUCTION TO BASIC

لغة « بازيك »

BASIC LANGUAGE

1 - مقدمة :

تعتبر لغة بازيك BASIC من اللغات المتطورة ذات المستوى المرتفع (High level language) ، أي من اللغات التي لا تتعلق بنوعية الآلة وتكوينها الداخلي . وكلمة « بازيك » BASIC هي اختصار للجملّة :

Beginners All Purpose Symbolic Instruction Code

وقد ظهرت لأول مرة في سنة 1963-1964 نتيجة عمل مجموعة من الطلاب تحت إشراف البروفسور John Remmey والبروفسور Tomas Kurts ، في مدرسة Darmuth . وكانت مخصصة للمبتدئين في تعلم البرمجة ، لسهولة وبساطتها . ولكن هذه اللغة ما لبثت أن أخذت بالتطور نتيجة الاهتمام الزائد بها من قبل منتجي الآلات الحاسبة وخاصة شركات الميكروكومبيوتر . وبنتيجة زيادة العمل بنظام الوقت المُقسَّم Time sharing من قبل الباحثين والمخبريين . وتُعتبر لغة « بازيك » بالإضافة للغة فورتران وباسكال من اللغات الشائعة الاستعمال والأكثر استعمالاً وشيوعاً في الأوساط الجامعية والمدرسية ، ويقوم بتدريسها في أكثر الجامعات ، نظراً لمقدرتها على حلّ المسائل الرياضية والهندسية والعلمية بشكل عام ، وبساطتها وسهولة فهمها واستعمالها . وتُستعمل هذه اللغة بشكل خاص في نظام الوقت الحالي (Real time) ومع الآلات الصغيرة (Micro Computer) . لذلك نلاحظ أن أكثر المبرمجين لهذه اللغة هي الشركات التي تنتج الآلات الحاسبة الصغيرة .

ولقد ظهرت كثيراً من المفسرات المخزنة في الذاكرة الثابتة للحاسب والتي تستطيع معالجة وتكجمة البرنامج مباشرة إلى لغة المكنة مما يعني سهولة في الاستعمال .

وأهم النماذج المستعملة والمسوّقة هي التالية :

مصرف : compiler مفسرة : Interpreter

- M BASIC - هي الصيغة الأساسية للبازيك من إنتاج Microsoft ومنها جرى عدة توسيعات أدت إلى ظهور نماذج عديدة من هذه اللغة . وتحتوي على أهم التعليمات الأساسية في هذه اللغة .

- الصيغة MS-BASIC ، هي نموذج مُوسَّع من الصيغة M BASIC والعاملة بإشراف النظام CPM 180 . والتوسيعات والإضافات المقررة هي حول الدوال \$ DATE , TIME .

- المؤول GW-BASIC وهو عبارة ن صيغة مُوسَّعة من MS-BASIC تحتوي على الإضافات الأساسية التالية :

- تنقيح كامل للشاشة .

- توسيعات وإضافات للرسوم البيانية (graphics ext.) .

- توسيعات وإضافات حول الموسيقى (music ext.) .

- تنظيم وإدارة الخط التسلسلي (Serial line control) .

ولقد عرف هذا المؤول نجاحاً كبيراً وبالتحديد في الولايات المتحدة واليابان ، مع العلم بأنه لا يستطيع العمل إلا على ميكروكومبيوتر مُجهَّز بشاشة للرسوم (للرسوم البيانية) وبذاكرة من نوع «bit map» .

- المصرَّف BAS COM ، وهو موجه بشكل أساسي للبرامج التطبيقية الكبيرة والمعقدة ، التي يرغب المالك بحفظ الصيغة الأصلية للبرنامج مع إصدار نسخة سريعة التنفيذ .

التشكيل اللغوي لهذا المصرَّف يشبه التشكيل اللغوي للمؤول MS-basic مع فارق وحيد هو في عدم إمكان إستعمال التنظيم الفرضي الديناميكي للذاكرة (Virtual memory) .

- المصرَّف بازيك للتجارة (Busniss Basic Compiler) وهو عبارة عن مصرَّف بازيك مُوَّه ومُخصَّص للعمل بالتمثيل العشري للأعداد وبدقة مؤلفة من 14 رقماً بدلالة عالية . (14 significant digits) وهو شبيه بالمصرَّف C BASIC الذي نشرته شركة (Digital research) .

- المؤول P-BASIC الذي يستطيع قبول وإستقبال نفس البرامج الأصلية للمؤول BASIC-86 من شركة microsoft . ويحتوي على مُحلِّل لغوي لتحليل الأسطر

قبل ضرب الرمز الأخير لعودة السطر (CR) .

- المؤول S-BASIC من إنتاج الشركة الفرنسية Siros . ويعمل بدقة مضاعفة ، ويستقبل متحولات مركزية وعامة ، ويحتوي على تعليمات للرسم البيانية .

- المؤول MEM BASIC . ويحتوي على مُنقح (Editor) كامل للشاشة ، ويسمح بفتح عدة فسحات في الشاشة ، وبالبرمجة التركيبية (Structural Programming) وذلك بواسطة التعليمات DO ، LOOP ، SELECT ، CASE ، كما يسمح باستعمال وسم أبجدي للتعليمات . يعمل هذا المؤول بإشراف النظام PC-DOS و MS-DOS وذلك بالاتصال بالنظام MEM-DOS .

- TRVE BASIC ، الذي صدر بشكل مُصرّف مع محيط من نوع مؤول . ويحتوي على مُنقح كامل للشاشة . وعلى تعليمات للرسم والجداول ولإدارة الأخطاء . أرقام الأسطر والتعليمات هي غير ضرورية ، وبالإمكان تقسيم الشاشة إلى فسحة للبرنامج وفسحة للتنفيذ ، أما الحسابات فيمكن أن تتم بالنظام الثنائي وبفاصلة مُتحرّكة أو بدقة مضاعفة .

- الخ . .

2 - أسس لغة بازيك

البرنامج بلغة البازيك ، عبارة عن مجموعة من الرموز ، تتألف من أرقام وأحرف : الأحرف اللاتينية الكبيرة (A ÷ Z) والأرقام العربية (0 ÷ 9) ، والرموز الخاصة التالية :

+	Plus زائد
-	Minus ناقص
↑ ↑	Up Arrow متجه عامودي
/	Slash خط منحنى
*	Asterisk نجمة
()	Left and right parenthese أهلة (هلال)
,	Comma فاصلة
.	Decimal point فاصلة عشرية
&	Ampers and
;	Semi colon نقطة فاصلة
:	Colon نقطتين عاموديتين
	Blank فراغ بياض
'	Single quote أبوستروف
"	Double Quote
= < >	Essor equal أصغر أو يساوي
=	Equal يساوي
>	Great أكبر
> = >	Great or equal أكبر أو يساوي
≠	different يختلف
!	علامة تعجب
\$	Dolars Sign دولار
@	Atsign
#	Number sign

تؤلف كل مجموعة من الرموز المتتالية والمتصلة ببعضها ، والتي تحتوي على فراغات أو بدون فراغات (Blank) كلمات مختلفة (Word) . كل مجموعة من هذه الكلمات المنفصلة الواحدة عن الأخرى بفراغات ، تؤلف جملاً أو أوامر (Statement) بلغة بازيك . ويُفهم من كل جملة أمراً من الأوامر تقوم الآلة بتنفيذه . فالكلمات إذاً ، هي عبارة عن متحولات وأعداد وأوامر عمليات (Command) .

وتحتوي لغة بازيك على أربعة أنواع من الأوامر :

- 1 - أوامر حسابية : Arithmetic Statement .
- 2 - أوامر إدارية : Control Statement .
- 3 - أوامر الإدخال والإخراج : Input / Output Command .
- 4 - أوامر التخزين والمواصفات التي تقوم بالعمل مع الذاكرة : Specification Statement .

ولكتابة برنامج معين بلغة البازيك ، يجب أن نحافظ على الأمور التالية :

- يحتوي كل سطر على تعليمة واحدة . لكن هناك آلات تسمح بكتابة أكثر من تعليمة واحدة على نفس السطر على أن تكون منفصلة عن بعضها بالرمز (:) .
- كل برنامج بلغة بازيك يجب أن ينتهي بالأمر END .
- لكل أمر من الأوامر أو تعليمة من التعليمات أو سطر من الأسطر ، رقماً واحداً . على أن تبدأ الأرقام بالعدد 1 وتنتهي بالعدد 99999 .
- تنفيذ التعليمات يتم حسب ترتيبها الرقمي .
- يجب أن لا يؤخذ بعين الاعتبار ، الفراغات الموجودة بين الكلمات ، والمتحولات والأعداد .

- يجب أن لا تحتوي الأعداد على فراغات .

- يمكن إدخال أي تعليمة من التعليمات إلى أي مكان في البرنامج ، على أن يتم ترقيمه بشكل لا يتطابق الرقم الجديد مع أي رقم آخر في نفس البرنامج .
- نستعمل الأمر NEW عند كتابة برنامج جديد .
- مثل : أكتب البرنامج الذي يقوم بحل المعادلة التالية :

$$\sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}$$

$$a_1 = 2, a_2 = 8, a_3 = 21, a_4 = 17$$

```
10      REM      PROGRAM
20      DIM      A[ 4]
30      LET      X= 0
40      FOR      I= 1 TO 4
```

```

50      READ A [ I ]
60      LET    X= X+ A[ I]
70      NEXT  I
80      LET    S= SQR (X)
90      PRINT
100     DATA  2,8,21,17
110     END

```

RUN

النتيجة : 28.2489

3 - أنواع المعطيات البيانية Data information

3.1 - الأعداد

تختلف الأعداد المستعملة بطولها حسب نوع الآلة ، مثلاً : طول العدد المستعمل في الميكروكومبيوتر Coumpucorp هو 11 رقماً ، وتتراوح الأعداد بين $1 \times 10E - 38$ إلى $1 \times 10E + 38$. أما الميكروكومبيوتر Motorola 8 bits فتتراوح الأعداد المستعملة بلغة البازيك بين $1.0E - 99$ إلى $9.99999999E + 99$. أما بالنسبة للآلة الكبيرة (Largecomputer) ، فطول الأعداد المستعملة بلغة البازيك هو نفسه المسموح به في بقية اللغات أي بطول 32 bits أو 64 bits وذلك حسب نوع الكومبيوتر وتصميمه ، وطول المراصف الداخلية للحاسب المركزي المستعملة (Reigister) . ويتم إدخال الأعداد إلى الآلة الكبيرة عادة بواسطة البطاقات المثقوبة مباشرة بواسطة لوحة الملاصق أو بغيرها من الطرق . أما بالنسبة للميكروكومبيوتر فالمعلومات يتم إدخالها بواسطة لوحة ملاصق الشاشة الكاثودية وبواسطة عدد من أوامر بازيك مثل DATA و INPUT . أما الإخراج فيتم بواسطة التعليمات PRINT أو PRINT USING .

3.2 - الرموز STRING

المعلومات الرمزية عبارة عن لائحة أو مجموعة متلاحقة من الرموز

سجل ، مرصف : Register لوحة ملاصق : Key bord

الأبجدية والحاسبة مثلاً :

LEBANON IS MY COUNTRY

العدد الرمزي يمكن أن يحتوي حتى 255 رمزاً.

3.3 - الثوابت Constants

أ - الثوابت العددية

وتتألف من الأرقام العربية 0 ÷ 9 وبإشارة حسابية + ، - وطولها يخضع لتصميم الآلة ونوعيتها . وطول الكلمة المستعملة والمعالجة كوحدة معلوماتية .

مثلاً : 60045.45 - 5E11 \$ + 5

1.60ZE- 19 + 1.60ZE+ 19

والثوابت العددية هي :

- عشرية صحيحة Integer

مثلاً : 600

342678

10452

ب - عشرية بفاصلة ثابتة FIXED DECIMAL

مثلاً : 600.05

10452.00

- 104.52

ج - بفاصلة عشرية متحركة FLOAT DECIMAL

ويتألف هذا العدد من عددين أحدهم يدعى جزء عشري MANTISSE

والآخر يُدعى أس EXPONENT على الشكل التالي :

$$\pm M \times E \pm n = \pm M \times 10 + n$$

مثلاً : 1.60ZE- 19 ، 1.60ZE+ 19

د - الثوابت السادس عشرية .

- وتتألف من الأرقام العربية 0+9 والأحرف اللاتينية A, B, C, D, E, F .
ويجب أن يكون العدد السادس عشري مسبقاً بالرمز μ حتى تفهمه الآلة .
مثلاً : μEA
 $\mu F4BC0$

هـ- الثوابت الثنائية

وتتألف من الرقمين 0 و1

مثلاً : 111011

00100

وهذه الثوابت يمكن أن تكون بفاصلة عشرية ثابتة ومتحولة أو صحيحة .

و- الثوابت الرمزية Constant string

وهي على الشكل التالي : Constant string

أي موجودة بداخل (" ") : ويتم إدخالها وإخراجها دون أي تغيير فيها وتتألف مثلاً : من مجموعة من السمات .

“ LEBANON IS MY COUNTRY ”

3.4 - المتحولات VARIABLES

على خلاف اللغات الرمزية ، كلفة المؤول ، حيث يتوجب على المبرمج أن يحفظ ، ويعرف عناوين القيم الثابتة والمتحولة في الذاكرة للعمل بها ، وإجراء المعادلات والعمليات الرياضية والحسابية وغير ذلك . فاللغات ذات المستوى المرتفع والمتطورة كلفة « بازيك » تسمح للمبرمج بعدم معرفة وحفظ هذه العناوين ، بل بتسمية هذه القيم المتحولة والثابتة ، العددية والرمزية ، وحفظ أسماءها فقط . وتقوم الآلة بواسطة نظام التشغيل (Operation system) بحفظ مكان في الذاكرة لهذه الأسماء التي تُعبر عن القيم المتحولة وحفظ عنوان مكانها ، وعند الحاجة يكتب المبرمج اسم المتحولة فقط فتقوم الآلة باستدعائها من الذاكرة دون أن يكون للمبرمج علماً بمكانها أو عنوانها . هذه الأسماء تدعى متحولات (Variables) .

مثلاً : $AA = LEBANON$

مؤول : Assembler

المتحولة AA هي رمزية وقيمتها هي LEBANON

$$A = 10452$$

المتحولة A هي عددية (رقمية) وقيمتها 10452 .

وأثناء تنفيذ البرنامج ، تتغير قيمة المتحولات حسب العمليات المنفذة في البرنامج ، وهذا معناه إن مخزون الخلايا المحفوظة لإسم المتحولة يتغير حسب المعادلات والعمليات المنفذة في البرنامج . ولكن عند البدء بتنفيذ البرنامج بواسطة الأمر (RUN) ، تقوم الآلة بإعطاء قيم أولية للمتحولات ، « صفر » للمتحولات العددية و« فراغ » للمتحولات الرمزية .

وهناك نوعان من المتحولات :

أ - المتحولات العددية ، وهي عبارة عن أسماء للأعداد .

ب - المتحولات الرمزية وهي عبارة عن أسماء للرموز والسمات أو للمتحولات الرمزية .

ومن غير المسموح به أن تبدأ أسماء المتحولات بالأرقام ، بل يجب أن تبدأ بحرف ، ويمكن أن يكون متبوعاً بحرف أو بأرقام أخرى .

مثلاً :

A 20, C, AB, A, B, C, ...

هي عبارة عن أسماء لمتحولات عددية .

أما أسماء المتحولات الرمزية فيجب أن تكون متبوعة بالرمز § . مثلاً :

A § , ZZ § , T § , ...

كما ويجب أن لا يتعدى طول المتحولة الرمزية 18 سمة للمتحولة الواحدة . وهكذا عندما تقرأ الآلة هذا النوع من المتحولات حيث يكون إسم المتحولة متبوعاً بالرمز § ، فتدرك فوراً إن المتحولة هي رمزية فتحفظ لها مكاناً بالذاكرة بطول 18 بايت (18 bytes) ، كما وتحفظ عنوانها لاستدعائها عند الحاجة .

3.5 - الجداول (ARRAY) والمصفوفات .

كل مجموعة من المتحولات من نفس النوع ونفس القياس والجنس تحفظ في الذاكرة كجدول من المتحولات وفي أماكن متقاربة الواحدة تلو الأخرى .

لكل جدول من هذه الجداول إسماً واحداً كأسماء المتحولات مسبوقاً بعدداً

يدلّ على عدد المتحولات في الجدول .
Array (n₁, n₂, n₃)

Array - اسم الجدول .

n₁, n₂, n₃ - أعداد تدل على عدد المتحولات بداخل الجدول .

مثلاً : LEB (10452,2)

جدول مربع باسم LEB . ويحتوي على 2 × 10452 متحولة ، موزعة على 10452 سطرًا وعمودين .

مثلاً : LE (10)

عبارة عن جدول أو لائحة بطول عشرة متحولات ، أي يحتوي على عشرة متحولات مثلاً : L (4,5,2)

جدول مثلث يحتوي على 40 = 2 × 5 × 4 متحولة .

والتصريح عن جداول المتحولات يتم بواسطة الأمر DIM متبوع باسم الجدول أو المصفوفة وبأبعادها .

مثلاً : DIM . LE (10, 45, 2)

وبواسطة هذا الأمر نقوم بالتصريح عن جدول باسم LE وبأبعاد 10, 45, 2 .
ويحتوي على 2 × 45 × 10 = 900 موقعاً من الذاكرة ويتسع إلى 900 متحولة عددية أو رمزية حسب نوع المتحولات والتصريح عنها .

4 - المعادلات EXPRESSIONS

المعادلات عبارة عن لائحة من المتحولات والثوابت والدالات ، متصلة ببعضها بإشارات حسابية (للجمع أو الضرب أو القسمة . .) ، أو منطقية (AND, NOT, ...) وغير ذلك . ويمكن أن تكون جبرية أو منطقية .

مثلاً : 2 + 3 ;

(A + 19.23) (SIN (X) + 5)

أ - الإشارات الحسابية المستعملة في المعادلات وأولوياتها :

- الرفع ب (**) EXPONENT

ولها أولوية في عملية الحساب .

مثلاً : $2 * * 3 + 5 = 2^3 + 5$

- الإشارة السلبية (-) NEGATIVE

مثلاً : $- 4 ، - a 1$

- الضرب (*) ، والقسمة (/) Multiplication and division

- الجمع (+) أو الطرح (-) Addition and Soustraction

مثلاً : $2 * 3 + 4 / 5 - 6 = \frac{2^3 + 4}{5 - 6}$

$A * * 3 + 5 / \sin(X * * 3) - 6 * A = \frac{A^3 + 5}{\sin(X^3) - 6 A}$

ب - المعادلات الشرطية والمقارنة

ترتبط المعادلات الشرطية بإشارات للمقارنة ، ولها قيمتان تعادل : (1 -)
أو (0) .

وبواسطة المعادلات الشرطية ، وإشارات المقارنة ، يمكننا مقارنة المتحولات العددية والرمزية ببعضها ، وتحديد تنفيذ شروط معينة أو حوادث قد تحدث أثناء عمل البرنامج .

والإشارات المستعملة في المقارنة هي :

<	أصغر
>	أكبر
<> ، ><	لا يعادل
=	يعادل
<= ، = <	أصغر أو يعادل
>= ، = >	أكبر أو يعادل

وهكذا ، إذا كانت المعادلة الشرطية نافذة أو حقيقة فقيمتها ستكون (1 -)

وإلا فستكون قيمتها (0) .

مثلاً :

النتيجة	معادلة شرطية
(حقيقة) - 1	$1 < 2$
(غير حقيقة) 0	$1 > 2$
(حقيقة) - 1	$1 < > 2$
(غير حقيقة) 0	$1 > = 2$

عند مقارنة المتحولات الرمزية ، يجب تحويل هذه الأخيرة إلى أعداد ثنائية معادلة للرموز ، وعند ذلك مقارنتها .

مثلاً :

(غير حقيقة) 0	"A" = "B"
(حقيقة) - 1	"A" < "B"

مثال على المعادلات الشرطية :

$$A * 4 + 3 \sin(X) > A / B + 4 B$$

ولحل المعادلة الشرطية ، تقوم الآلة بحساب أجزاء المعادلة ، وإيجاد قيمة كل منها وبالتالي مقارنتهم .

ج - المعادلات المنطقية Logic expression

وتتألف من أعداد أو متحولات مرتبطة ببعضها بإشارات منطقية, AND, OR, NOT . ولتنفيذ المعادلة المنطقية تقوم الآلة بتحويل الأعداد أو قيمة المتحولات إلى أعداد ثنائية بطول 16 بته (16 bits) بالنظام الثنائي ، وعند ذلك تقوم بإجراء (حساب) المعادلة المنطقية .

مثلاً :

نتيجة	معادلة
$2 = N$	$N = 14 \text{ AND } 18$
$30 = M$	$M = 14 \text{ OR } 18$
$- 15 = C$	$C = \text{NOT } 14$

مثال على المعادلات المنطقية :

$$M = (A * B) \text{ AND } (C * D / 4 + C)$$

5 - أوامر الإدخال والإخراج .

5.1 - الأمر « أدخل » . INPUT

هذا الأمر يُمكِّن المبرمج من تلقيم الآلة والبرنامج بالمعطيات الثابتة (Data) والرمزية (CHARACTER) ، الضرورية للحصول على النتائج . عملية الإدخال تتم عادة بواسطة التعليمة INPUT ومباشرة من لوحة الملامس (CRT Key board) التابعة للكمبيوتر .

وشكل الأمر INPUT هو التالي :

< Ligne NUMBER > { < FILE NUMBER > , { ... < Device NAME > ...

... { (< Longueur >) } , { { < Symbol String > ; } < Lists of variables >

المتحولات الداخلة يجب أن تكون منفصلة عن بعضها بفواصل .

10 INPUT A

أمثلة :

20 INPUT A " CUSTOMER BALANCE " ; A

بواسطة التعليمة رقم 20 يتم إدخال متحولة رمزية هي Customer Balance ومتحولة عددية هي A .

30 INPUT # 7 , A , B , C

يتم إدخال ثلاثة متحولات هي A , B , C وعدد عشري هو 7 .

40 INPUT # 10452 , "LEBANON" , BE \$

يتم إدخال العدد 10452 والثابتة الرمزية LEBANON والمتحولة الرمزية

. BE

عند تنفيذ البرنامج ، وللدرد على الأمر INPUT يقوم المُفسِّر للغة البازيك بالإجابة بالرمز ؟ ، عند ذلك يجب على المبرمج أن يقوم بإدخال المعطيات المعادلة

للمتحوّلات ، حسب ترتيبها في لائحة المتحوّلات .

مثلاً : `20 INPUT A, B, C, "LEBANON", D` §

فيجيب البازيك بالرمز ؟

عند ذلك يجب إدخال قيمة المتحوّلات A, B, C . مثلاً :

10452 , 452 , 52 , BEIRUT

عند ذلك ستأخذ المتحوّلات القيم التالية :

A= 10452, B= 452, C= 52, D= BEIRUT

أما الثابتة الرمزية LEBANON فيتم إدخالها وإخراجها دون تغيير .

مثل آخر : `30 INPUT "LEBANON", D` §

فيجيب البازيك بـ : ? LEBANON

عند ذلك يجب إدخال قيمة المتحوّلة الرمزية D مثلاً : BEIRUT (CR)

فتأخذ المتحوّلة الرمزية D الثابتة BEIRUT . وتكون النتيجة عند

الإخراج : LEBANON BEIRUT .

5.2 INPUT LINE

يستعمل عادة لإدخال السمات . وبواسطة يتم إدخال أسطر معلوماتية من

السمات إلى البرنامج .

أمثلة : `10 INPUT LINE "LEBANON", A`

هذا الأمر يقوم بإدخال السمات من السجل رقم خمسة ، وتلقينها الى

المتحوّلة الرمزية B .

`10 INPUT LINE 5, B` §

هذا الأمر يقوم بإدخال الرموز من الفايل رقم خمسة ، وتلقينهم للمتحوّلة

الرمزية B .

5.3 الأمر التعليمية FETCH

يُستعمل لإدخال المتحوّلات الرمزية بطول 1 بايت (one liste

. character string)

وشكل التعليمة هو :

FETCH < FILENUMBER > , < Liste of variables string > ,
< Logic device name >

مثلاً :

FETCH § 5 , A § , B §

يتم إدخال الثوابت الرمزية بطول بايتة (Byte) واحدة من السجل رقم خمسة وإعطائها للمتحولات A و B .
عند تنفيذ التعليمة ، يتم إدخال البايت الأول إلى المتحولة الرمزية الأولى في لائحة المتحولات .

5.4 - الكتابة والقراءة READING and WRITING

أ - التعليمات READ-DATA

لتلقي المتحولات بالثوابت ، ولتخزين هذه الثوابت في الذاكرة ، نستعمل الأمر READ مسبقاً أو متبوعاً بلائحة من المتحولات :

READ V_1, V_2, V_3, \dots

V_3, V_2, V_1 - عبارة عن متحولات مصرّح عنها سابقاً ، والأمر DATA مسبق بلائحة من الثوابت :

DATA K_1, K_2, K_3, \dots

بعد تنفيذ هذين الأمرين ستأخذ المتحولات V_1, V_2, V_3, \dots القيم

التالية :

$$V_1 = K_1$$

$$V_2 = K_2$$

$$V_3 = K_3$$

مثلاً :

10 READ A, B, C

```

20    LET      D= A+ B+ C
30    PRINT    D
40    DATA    10, 14, 26

```

بعد تنفيذ هذا البرنامج سيقوم الكمبيوتر بإخراج القيم العددية التالية للمتحولات على الشكل التالي :

A= 10

B= 14

C= 26

وبعد ذلك تتم عملية الجمع للحصول على النتيجة المطلوبة D .

مثلاً :

```

10    DATA    1, TWO, "THREE"
20    DATA    4, 56, 10452

```

لإدخال الفراغ بداخل الثابتة الرمزية يجب استعمال () . مثلاً :

```

10    DATA    "BEIRUT"

```

الرمز الأول في المتحولة هو فراغ أما إذا كتبنا الأمر على الشكل التالي :

```

10    DATA    "BEIRUT"

```

فالرمز الأول هو B . يُستعمل ذلك لتوزيع المعلومات على السطر في محاولة لإخراجها بشكل لائق للقراءة وإحداث فراغات بين الكلمات .

أمثلة على استعمال التعليمات DATA و READ

```

110   DATA    10, 20, 30, 10452, "LEBANON"

```

.....

```

140   READ     A, B, C, L, BE

```

بنتيجة تنفيذ هذه التعليمات ستحصل كل متحولة على قيمة ثابتة على

الشكل التالي :

A= 10

B= 20

C= 30

L= 10452

BE= LEBANON

القواعد العامة لاستعمال التعليمات DATA-READ

- إذا كان الأمر الأول READ ، يحتوي على عدد من المتحولات أقل من الثوابت الموجودة في الأمر DATA ، فالأمر التالي READ يبدأ بقراءة الثوابت من DATA ومن حيث انتهى الأمر الأول .

- إذا كان البرنامج يحتوي على عدد من الأوامر DATA ، فالثوابت فيهم ستؤلف لائحة متواصلة . وكل أمر من الأوامر READ يبدأ بقراءة الثوابت من حيث انتهى الأمر السابق له READ . مثلاً :

DATA 10, 4, 52

DATA BEIRUT

DATA LEBANON, 10, 452

هذه الأوامر الثلاثة تعادل :

DATA 10, 4, 52, BEIRUT, LEBANON, 10, 452

- إذا كانت المتحولة في READ رمزية ، فستعتبر الآلة ، الثابتة المخصصة لها رمزية ، ولن تأخذ بعين الاعتبار السمات (") .

- إذا كانت المتحولة عددية ، فيجب أن تكون الثابتة المخصصة لها في DATA عددية ، وإلا سيقوم الكمبيوتر بإعطاء إشارة إلى حدوث خطأ على الشكل التالي :

SINTAX ERROR AT LINE XXX

رقم السطر xxx هو رقم التعليمة DATA حيث يوجد خطأ في الثابتة .

- القراءة بواسطة READ خارج حدود الثوابت في المعطيات DATA غير مسموح به وسيقوم الكمبيوتر بإخراج الإشارة التالية لذلك :

OUT OF DATA AT LINE XXX

xxx رقم السطر أو التعليمة حيث يوجد الخطأ .
مثلاً :

10 DATA 1,2,7,THREE,"BEIRUT",5.01

20 READ A,B\$

فستحصل المتحولة العددية A على القيمة 1 ، أما المتحولة الرمزية B فستحصل على السمات الثلاثة 207 (كرموز وليس كأرقام) .

30 READ C\$

المتحولة C ستأخذ الثابتة الرمزية THREE

40 READ LE\$,E

عند ذلك ستأخذ :

LE= BEIRUT

E= 5.01

وإذا وُجد في البرنامج أمراً آخر من نوع READ ، مثلاً :

50 READ F

فستقوم الآلة بإعطاء إشارة إلى حدوث خطأ .

OUT OF DATA AT LINE 50

لأن المعطيات DATA لا تحتوي على ثوابت إضافية للمتحولة F .

5.5 - الأمر RESTORE

هذا الأمر يستعمل لإعادة قراءة الثوابت من الأمر DATA إلى أول اللائحة .

RESTORE LINE NUMBER

LINE NUMBER - رقم الأمر DATA ، أو أول أمر DATA يتلو هذا

الرقم .
مثلاً :

10 DATA 1,2,3

20 DATA 4,5,6

30 READ A,B

المتحولة A ستأخذ القيمة 1 ، والمتحولة B ستأخذ القيمة 2 .

40 RESTORE 20

50 READ A, B

الآن المتحولة A ستأخذ القيمة 4 والمتحولة B القيمة 5 .

60 RESTORE 10

70 READ A, B

 A = 1

 B = 2

10 DATA 1, 2, د أي إعادة استعمال الأمر

5.6 - إعطاء قيمة للمتحويلات (LET)

الأمر LET يعطي قيمة ثابتة لمتحولة معينة والشكل العام لهذا الأمر هو :

{ LET } < variable > = < expression >

variable : إسم المتحولة ، expression : تعبير جبري .

ولكن يجب أن تكون المتحولة والمعادلة من نفس النوع .

 LET A = 10452.0000 مثلاً :

 LET LE = "LEBANON"

 LET A = (2 + (3.8 / A * 2)) / (A - 1)

 LET A = SIN (M * 2 / K + 2)

 LET M = "BEIRUT"

الأمر LET يُستعمل لإعطاء قيم ثابتة للمتحويلات بداخل البرنامج كالأوامر

READ-DATA أو INPUT مع الفرق بأنه يستطيع إعطاء قيمة معادلة رياضية

متكاملة لمتحولة من المتحويلات .

110 DATA 1, 2, 3 مثلاً :

120 DATA 4, 5, 6

130 READ A, B, C, D, E,

هذه الأوامر تعادل :

```
110 LET A= 1
120 LET B= 2
130 LET C= 3
140 LET D= 4
150 LET E= 5
160 LET F= 6
```

ولو افترضنا الآن أن عدداً من الثوابت مُسجلاً في بنكاً للمعطيات (DATA BANK) على الشكل التالي :

. 10452, 60, 40, 25

فالبرنامج التالي :

10	READ	A, B	
20	LET	C= A+ B	
30	READ	D, E	DATA BANK
40	LET	F= C + D + F	40
50	PRINT	F	25
60	DATA	25	40
70	DATA	40, 60	60
80	DATA	10452	10452
90	END		

الأمر الأول READ يقرأ الثوابت الأولى 40, 25 . ومن ثم يقوم البرنامج بجمع المتحولات A و B . بعد ذلك بواسطة READ D, F يقرأ البرنامج الثوابت التالية 60 ، 10452 . . . هذا البرنامج يمكن كتابته على الشكل التالي بعد جمع جميع الثوابت في أمر واحد :

```
10 READ A, B, D, F
20 LET C= A+ B
30 LET F= C+ D+ E
```



```
40 PRINT F
50 DATA 25, 40, 60, 10452
```

7 - الشروحات COMMENTAR

بإستطاعة المبرمج إضافة الشروحات والملاحظات إلى برنامجهِ دون التأثير في مجراه . ويتم ذلك بواسطة التعليمة REM ، أو بواسطة ابوستروف (') على الشكل التالي :

```
10 REM / COMMENTAR, REMARKS /
20 COMMAND 'REMARK
```

مثلاً : تقرأ الآلة هذه الشروحات وتُخرجها دون التعليق عليها والتأثر بها .

```
10 REM BEIRUT IS THE CAPITAL OF LEBANON
20 A = A + 1 'INCREMENT POINTER
30 'BEIRUT IS THE CAPITAL OF LEBANON
```

8 - الأمر DIM و KIL

- نستعمل الأمر DIM للتصريح عن الجداول وأبعادها . على الشكل التالي :

DIM NAME OF ARRAY (DIMENSION)

DIM (أبعاد الجدول) < اسم الجدول >

مثلاً :

DIM A 1(4, 3)

الجدول ذو الاسم A 1 يحتوي على 3 × 4 متحولة ، وعندما تقرأ الآلة هذه التعليمة ستقوم بحفظ مكان في الذاكرة لهذه المتحولات .

- بواسطة DIM يمكننا التصريح عن عدد غير محدد من الجداول :

مثلاً :

DIM M (25) , C \$ (10, 15) , D (5, 3)

الجدول الأول يحتوي على 26 متحولة هي : M 0 , M 1 , ... M 25

الجدول الثاني C يحتوي على 16 × 11 = 176 متحولة رمزية هي

C 176... C 2 , C 1 , C 0

KILL_

نستعمل الأمر التعليمة KILL لتحرير المكان المحفوظ في الذاكرة للجدول المُسمى في الأمر .

KILL < Name of array > { , < name of array > } ...

بعد تنفيذ الأمر KILL ستقوم الآلة بمحو الجدول ذو الإسم name من الذاكرة ، لاستعمال مكانه لجداول أو لمتحولات أخرى .
مثلاً :

10 DIM A (15, 20)

.....

100 KILL A

110 DIM A (12, 30)

.....

200 KILL A

210 DIM A (11, 50)

9 - الأوامر STOP وEND

END_ 9.1

END - يعيد مهمة المراقبة إلى برنامج المراقب monitor أو إلى المُفسِّرة (interpretor) للغة البازيك ، ويغلق الفايل المسجل على الأسطوانة المغناطيسية (DISK FILE) . بعد هذا الأمر END ، لن يكون بإمكاننا متابعة تنفيذ البرنامج من موقعه ، لهذا يجب إعادة التنفيذ من البداية باستعمال RUN .

بعد الأمر END يجب البازيك بكلمة READY ، أي انه جاهز لتلقي الأوامر التالية .

STOP_ 9.2

STOP - يعيد مهمة المراقبة إلى البرنامج مراقب (monitor) ، ونجيب الآلة بـ :

STOP AT LINE XXX

READY

بإدخال كلمة CONT من لوحة ملامس الكمبيوتر ، سيقوم الأخير بمتابعة تنفيذ البرنامج من النقطة التي تتبع الأمر STOP .

نستعمل STOP عادة ، أثناء تصحيح الأخطاء في البرنامج (Program testing) وEND لإنهاء العمل به .

10 - الطباعة PRINTING

PRINT- 10- 1

يستخدم لإخراج النتائج (أعداداً ورموزاً) على جهاز خارجي (الشاشة التلفزيونية ، إسطوانة ، آلة الطباعة) والشكل العام لهذا الأمر هو :

PRINT { < Device logic name > , { < Liste of expressions, variables > } ; | ? |

| ? | < FILE NUMBER >

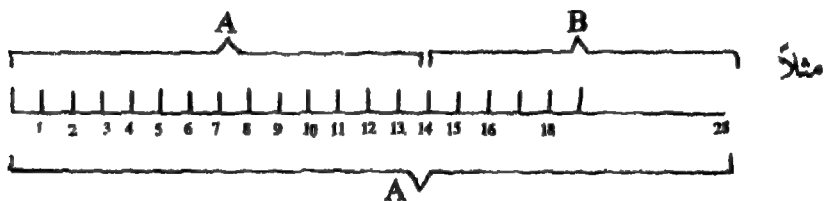
يمكن استعمال الرمز (؟) بدل الأمر PRINT في بعض الحالات وله نفس المفهوم بالنسبة للآلة .

لائحة المتحولات والمعادلات (Liste of expressions, variables) ، يمكن أن تحتوي على عدد غير محدود من المعادلات والمتحولات العددية أو الأبجديّة . عناصر اللائحة تفصل عن بعضها بفواصل (,) أو (:) .

مثلاً :

PRINT A, B, C

يتم إخراج كل عنصر من موقع معين على السطر برقم ينقسم على 14 .



PRINT A; B; C أما الأمر :

فعند الإخراج سيكون كل عنصر مسبوق أو متبوع بفراغات على السطر .

المعادلات الموجودة في « لائحة المعادلات » يمكن أن تكون :

- معادلات ، مثلاً : $(2 + 3 * A) / 4$

- متحولات عددية أو أبجعددية رمزية ، مثلاً : A Q \$ (17) أو A ^

- ثوابت عددية أو أبجعددية ، مثلاً : 10452 أو "JOHN"

- دوال ، مثلاً : TAB (15)

- كود مصرح عنه بالشكل CHR \$ ، مثلاً : CHR \$ (7)

وفي حالة عدم ذكر لائحة المعادلات أو المتحولات يتم الانتقال إلى السطر التالي ، ويترك سطرًا فارغًا .
مثلاً : لنفترض أن

C = "LEBANON" , B = 2, A = 10.5

فالأمر التالي :
PRINT A, B, C \$

سيقوم بطباعة وإخراج : LEBANON 2 10.5 على الشاشة الكاتودية ..

أما الأمر التالي :

PRINT "MY COUNTRYIS" ; C \$

فسيقوم بإخراج الجملة التالية : MY COUNTRY IS LEBANON

PRINT # PRINTR, A + B, A - B, (A + B) / (A - B)

عند تنفيذ هذه التعليمة ، ستقوم الآلة بإخراج المعطيات التالية على آلة الطباعة :

12.5 8.5 1.470588235

- مثلاً :
10 PRINT A; B; C;
20 PRINT D

التعليمة الأولى (10) ستقوم بطباعة المتحولات A, B, C على سطرٍ واحد دون الانتقال إلى السطر التالي ، ولذلك ستقوم التعليمة رقم 20 بطباعة المتحولة D على نفس السطر .

مثلاً : اكتب البرنامج الذي يحسب قيمة الطاقة حسب المعادلة التالية :

$$\text{POWER} = I^2 * R$$

كما يقوم بجدولة النتيجة لمجموعة من القيم الثابتة I و R .

الحل : يجب أن يحتوي السطر الأول وفي رأس الصفحة على المعلومات

POWER RESISTANCE CURRENT : التالية

موزعة على خانات ، على أن تحتوي كل خانة على قيمة المتحولة التي سنحصل عليها بعد تنفيذ البرنامج .

البرنامج :

• • • • •

10 READ R,I

```
20 PRINT "POWER", "RESISTANCE", "CURRENT"
```

30 LET P = (I * 2) * R

40 PRINT P,R,I

50 DATA 10, 2, 10, 4

60 END

RUN

40	10	2
160	10	4
...

بوجود الفاصلة بين المتحولات في الأمر PRINT ، تقوم الآلة بتقسيم السطر إلى خمسة خانات طول كل منها 10 مواقع ، (أي تستطيع أن تستوعب 10 سمات) ، وسيتم إخراج النتائج بشكل جدول ، فتظهر كل قيمة في الخانة المخصصة لها حسب التعليم رقم (20) .

- أما لو أردنا إخراج عدداً أكبر من النتائج أو المتحولات ، فكل خمسة متحولات ستُعرض على سطر واحد والباقي سيظهر على السطر التالي ، مثلاً :

```
70 PRINT A, B, C, D, E, F, G
```

فعلى الشاشة ستُعرض المتحولات على الشكل التالي :

```
A    B    C    D    E
F    G
```

```
10 PRINT " JEDDA"          مثلاً :
```

```
20 LET P= 10452
```

```
30 PRINT P
```

فسنحصل على النتيجة التالية :

JEDDA = 10452

أي سيتم طباعة المتحولتين على سطر واحد .

مسألة : إكتب البرنامج الذي يحسب محيط ومساحة دائرة بشعاع R يعادل 2163 .

الحل : من المعلوم إن محيط ومساحة الدائرة تساوي :

المحيط : $D = 2\pi R$

المساحة : $S = \pi R^2$

البرنامج :

```
10 READ R
```

```
20 LET D= 2* 3.141* R
```

```
30 S= R * 3.14159
```

```

40 PRINT "RADIUS= "; R; "DIAMETER= "; D;
50 PRINT " AREA= "; S
60 DATA 2163
70 END

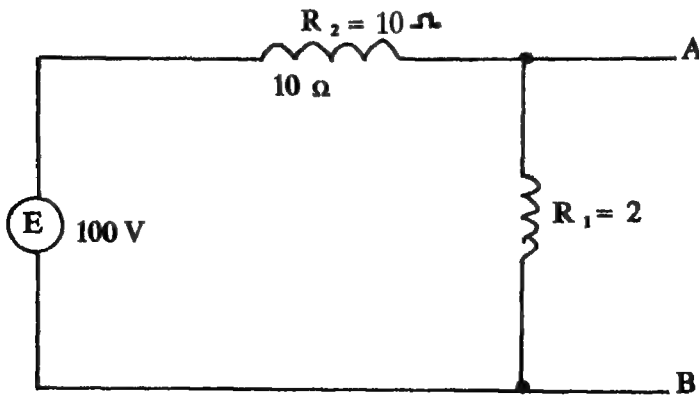
```

والنتيجة ستبدو على الشكل التالي :

RADIUS = 2163 DIAMETER = 4326 AREA = 1.4698E + 7

في هذا المثل سنرى ، أن الأعداد التي يزيد طولها عن تسعة أرقام ستطبع
بفاصلة متحركة (point flottante) .

مسألة :



اكتب البرنامج الذي يحسب قوة التيار الكهربائي في النقطة A حسب الدائرة
الكهربائية الموجودة على الصورة .

```

10 REM ELECTRIC CIRCUIT
20 READ E, R1, R2
30 LET A = E * R1 / (R1 + R2)
40 PRINT 'A= ' ; A
50 DATA 100, 50.2, 10
60 END

```

```

RUN
A= 83.3887

```

PRINT USING and FORMATING_ 10.2

يسمح الأمر PRINT USING بمراقبة شكل المتحولات الرقمية والرمزية ، عند إخراجها من الذاكرة ، ويقوم ب :

- التحكم بعدد الأرقام بعد الفاصلة العشرية .
- طباعة الاشارات الحسابية السلبية والإيجابية (- ، +) في بداية العدد .
- تعبئة الرموز الغير مستعملة في المتحولة بنجوم (*) .
- طبع الاشارة دولار (\$) في أقصى يسار العدد .
- طبع الفاصلة العشرية في المكان المحدد لها .
- طبع الأعداد وإخراجها بفاصلة متحركة ($\pm ME \pm n$) .

وشكل التعليمة PRINT USING هو :

a) **PRINT USING { < FILE NUMBER > , } < String format expression > < Device NAME >**
< EXPRESSION > ; < Liste of variables > { ; }

ب -

b) **PRINT USING < FILE number > , { < LINE number > ;**
< STRING FORMAT > , < Device NAME >
< List of variables, Expressions > { ; }

- () ، النقطة والفاصلة في نهاية الأمر ، تعني إن عملية الطباعة ستتابع على نفس السطر .

- FILE number ، رقم السجل أو الفايل حيث توجد المعلومات المطلوب إخراجها .

- Device NAME ، إسم الجهاز الذي سيتم إخراج المعلومات عليه ، (الطباعة أو الشاشة التلفزيونية CRT ، ...) .

- STRING ، الرموز المطلوب إخراجها .

- Liste of variables ، لائحة بالمتحولات والمعادلات المطلوب إخراجها

بواسطة PRINT USING .

STRING format expression ، عبارة عن معادلة أو متحولة رمزية ،
 بقيمة تدل على الشكل أو النسق الذي سيتم إخراج المعطيات (DATA) به .
 - في الأمر (ب) STRING FORMAT ، هي عبارة عن ثابتة رمزية بقيمة
 تدل على شكل المعلومات المخرجة ، ولا يجوز أن تكون مُسجَّلة بداخل أبوستروف
 (").

مثلاً :

```
PRINT USING " ## ### " ; A
```

```
PRINT USING # PRNTR, " ## . ### " ; A
```

```
A S= " ## . ### " : PRINT USING A $ ; A
```

تتألف الثابتة الشكلية الرمزية STRING FORMAT المُستعملة في التعليمة
 PRINT USING من :

أ - شكل الأعداد .

- يدل الرمز (#) على موقع لرقم (digit) معين .

- يدل الرمز (.) على موقع الفاصلة العشرية .

- يدل الرمز (-) على موقع الإشارة السلبية (-) التي تسجل على شمال العدد
 الشكلي الرمزي .

- إذا كان طول العدد كبيراً ولا يتسع له الشكل العددي المُسجل في STRING
 FORMAT والذي يتألف من الرموز # ، فسيتم طباعة العدد الحقيقي ألتام منه
 الموجود قبل الفاصلة العشرية مسبقاً بالرمز % .

أمثلة :

```
10 A = 2 / 3
```

```
20 PRINTING USING " ## . ## " ; A
```

سيتم إخراج العدد : 0.67

```
10 A = - 2 / 3
```

```
20 PRINTINT USING " ## . ## " ; A
```

سيتم إخراج وطباعة العدد - 0.67

```
10  A= - 200 / 3
20  PRINT USING "      "; A
```

سيتم إخراج العدد :

- 66.67 %

```
10  A= 2 / 3
20  PRINT USING "##.## + "; A
```

سيتم إخراج العدد + 0.67

```
10  A= + 2 / 3
20  PRINT USING " +##.##"; A
```

يتم إخراج العدد + 0.67

- لتعبئة الأماكن الغير مستعملة على يسار العدد بالنجوم ، يجب أن نكتب قبل الشكل (النسق) نجمتين (**) ، كل نجمة منها تدل على موقع لرقم معين ، ولا يجوز كتابة النجمة بعد الفاصلة العشرية .

أمثلة :

```
10  A= 100
10  PRINT USING " ** ##.##"; A
```

سيتم إخراج العدد 100.00 *

```
10  A= 1
20  PRINT USING " ** ##.##"; A
```

سيتم أو طباعة العدد : 1.00 ***

```
10  A= 100
20  PRINT USING " $ $ ##.##"; A
```

سيتم إخراج العدد : \$ 10.00

```
10  A= 1
20  PRINT USING " $ $ ##.##"; A
```

سيتم إخراج العدد : \$ 1.00

```
10  A= 10
20  PRINT USING " ** $ #.##"; A
```

سنحصل على : \$ 10.00 *

- لطباعة الفواصل (,) ، يجب كتابتها في الشكل أو النسق الرمزي (FORMAT STRING) ، ولكن على شمال الفاصلة العشرية ، كما ويمكن للفاصلة أن تعني مكاناً أو موقعاً لرقم معين .

مثلاً : 10 A= 10000

20 PRINT USING "##,#####";A

سيتم إخراج العدد التالي : 10,000.00

- لطباعة العدد بفاصلة متحركة ، يجب إدخال أربعة رموز (^) في نهاية الشكل الرمزي لعدد معين ، عند ذلك سيتم إخراج العدد على الشكل التالي :

< مرفوع ب > < العدد غير الصحيح > . < عدد صحيح >
< INTEGER > . < QUOTIENT > < EXPONENT >

مثلاً : 10 A= 1

20 PRINT USING "##.##^ ^ ^ ^";A

سيتم إخراج أو طباعة العدد التالي :

1.000E+ 00

30 PRINT USING "####.#### ^ ^ ^ ^";A

سيتم إخراج العدد 100.0000 E- 02

ب- المتحولات الرمزية .

ندلّ عليها بواسطة أبوستروف (') في الشكل الرمزي ، يتبعها عدد من الرموز الخاصة . أما الثوابت الرمزية الموجودة في الشكل الرمزي فيمكن إخراجها مع المعطيات (DATA) .

الرموز المستعملة في إخراج المتحولات الرمزية هي :

LL...L- ، لتركيز المتحولة الرمزية لجهة الشمال .

RR...R- ، لتركيز ومُعادلة المتحولة الرمزية لجهة اليمين .

CC...C- ، لتركيز المتحولة الرمزية حول نقطة أو موقع معين .

EE...E- ، لزيادة طول الحقل الذي سيتم إخراج المتحولة الرمزية عليه .

هذه الرموز يجب أن تكون مسبقة بأبوستروف (').

مثلاً :

```

10  INPUT A
20  PRINT " 1 2 3 4 5 6 7 8 9 0 "
30  PRINT USING 130; A $
40  PRINT USING 140;A $
50  PRINT USING 150;A $
60  PRINT USING 160;A $
70  GOTO 10
130 !' LLLL
140 !' RRRR
150 !' CCCC
160 !' EEEE

```

هذا البرنامج سيقوم بطباعة المعلومات التالية :

قيمة A \$:	x	xx	xxx
مواقع الطباعة	:	123456/890	123456/89	123456/89
معادلة وتركيز المتحولة لجهة الشمال L :	:	x	xx	xxx
معادلة وتركيز المتحولة لجهة اليمين :	:	x	xx	xxx
C	:	x	xx	xxx
E	:	x	xx	xxx
قيمة A \$:	xxxx	xxxxx	xxxxxx
	:	123456/890	13456/89	123456/89
L	:	xxxx	xxxxx	xxxxxx
R	:	xxxx	xxxxx	xxxxxx
C	:	xxxx	xxxxx	xxxxxx
E	:	xxxx	xxxxx	xxxxxx

امثلة :

1 - لطباعة الأعداد العشرية

```
10 REM DECIMAL FIELDS
20 : ###.## ##.#### #####.
30 READ X, Y, Z
40 PRINT USING 20, X, Y, Z
50 DATA 674.326, -6.143769, -68.3
60 END
```

RUN

674.33 -6.1438 -68

2 - طباعة الأعداد بفاصلة متحركة :

```
10 REM PRINT EXPONENTIAL NUMBERS
20 :# #### ↑↑↑ ##.## ↑↑↑ ## ↑↑↑
30 READ A, B, C
40 PRINT USING 20, A, B, C
50 DATA 627.423, « 374.689, 26.8
60 END
```

RUN

. 6274E+ 03 - 3.75E+ 2 3.E+ 01

3 - طباعة الرموز والسمات

```
10 REM PRINT LITERAL DATA
20 : "THE SQUARE ROOT OF 81 IS" ##
30 LET A= SQR(81)
40 PRINT USING 20, A
50 END
```

RUN

THE SQUARE ROOT OF 81 IS 9.

مسألة :

```
10  INPUT A
20  PRINT USING 120; A $
80  GOTO 10
120  ! THANK YOU, MR. 'E, FOR YOUR ORDER
```

الشكل الرمزي في الأمر 120 يحتوي على جملة متبوعة بالرمز E ، الذي يدل على أن الحقل المستعمل لإخراج المتحولة الرمزية يجب أن يكون طويلاً . وستأتي جملة أخرى بعد الرمز E .
ولو افترضنا إن المتحولة A \$ تعادل :

A \$= AHMED
A \$= JOHN
A \$= LIBAN

فسنحصل على النتيجة الرمزية التالية :

THANK YOU, MR. AHMED, FOR YOUR ORDER
THANK YOU , MR. JOHN, FOR YOUR ORDER
THANK YOU, MR. LIBAN, FOR YOUR ORDER

10.3 - الجدولة بواسطة TAB

يمكننا بواسطة TAB تحديد ومراقبة الفراغات الموجودة والمتروكة بين قيم المتحولات المطبوعة ، ومن إخراج أو طباعة العدد من موقع معين على السطر . وشكله هو .

TAB (< number , expression >)

expression أو Number عبارة عن عدد يدل على المكان الذي سنبدأ منه بالطباعة على السطر ، ولا يجوز أن يتعدى العدد (expression) Number 255
أو : $0 < \text{number, expression} < 255$

يمكننا استعمال TAB في لائحة المتحولات التابعة للتعليمية PRINT ، ولا يمكن استعماله أو إدخاله إلى المعادلات .
مثلاً :

```
10 PRINT #PRNTR, ".1111111111222222222233333333334"
20 PRINT #PRNTR, "123456789012345678901234567890123456
7890
30 PRINT #PRNTR, TAB (12); "x"; TAB (29); "x"
40 END
```

ستقوم الآلة بطباعة :

```
1111111111222222222233333333334
1234567890123456789012345678901234567890
      x              x
```

10.4 - طبع جداول المعطيات PRINT column of data

لإخراج المعطيات بشكل جداول وعلى أعمدة ، نستعمل الأمر الدالة TAB ، على الشكل التالي :

```
PRINT TAB (C 1); V 1; TAB (C 2); V 2; ...
```

PRINT - للطباعة والإخراج على الشاشة أو على الآلة المطبعية .
C 1 ، ... ، C 2 - رقم الأعمدة من السطر ، التي تدل على الموقع الذي سنبداً منه بإخراج أو طباعة المتحولات
V 1, V 2, ... - المتحولات التي سيتم إخراجها أو طباعتها .
مثلاً : معنا جدول يحتوي على ثلاثين عدداً . أكتب البرنامج الذي يقوم بطباعة هذا الجدول في ثلاثة أقسام وعلى العمود رقم 4 ، رقم 23 ، ورقم 33 .
...

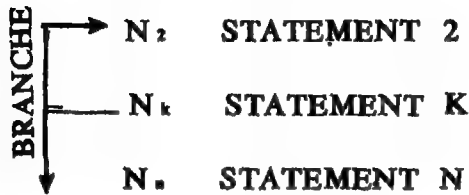
```
100 FOR K= 1 TO 30
200 PRINT TAB (3); K; TAB (22); K+ 10; TAB (33); K+ 20
300 NEXT K
400 END
RUN
```

عامود 4	عامود 23	عامود 33
1	11	21
2	12	22
3	13	23
4	14	24
5	15	25
6	16	26
7	17	27
8	18	28
9	19	29
10	20	30

11 أوامر التحكم والإدارة PROGRAM CONTROL STATEMENTS

بواسطة أوامر التحكم والإدارة يتم تغيير مجرى البرنامج المنطقي بشروط أو بدون شروط . كما يمكننا بواسطتها تنفيذ أقسام من البرامج أو حلقات من التعليمات (Loop) لمرات عديدة ، وتفاذي تكرار كتابتها .

N₁ STATEMENT 1



11.1 - الأمر GOTO بدون شرط

يقوم بتحويل مسار البرنامج بدون شرط إلى مكان آخر في البرنامج حسب الرقم المذكور بعده . وفي حالة عدم وجود هذا الرقم في البرنامج تقوم الآلة بإخراج إشارة تدل على حدوث خطأ فيه :

UNDEFINED STATEMENT AT LINE ×××

××× عبارة عن عدد يدل على رقم الأمر الغير الموجود في البرنامج ويتألف من ثلاثة أرقام .

مثلاً :
 100 GOTO 100
 805 GOTO 2050

مثلاً : لنفترض بأننا نريد أن نحسب قيمة الدالة :

$$y = x^3 + 7x$$

لعدد من القيم الثابتة x ، مثلاً : x=20, x=30, x=50, x=60 ، فالبرنامج هو :

```

...      ...      ...
10      LET      x= 20
15      LET      y= x↑3 + 7* x
20      PRINT    y
25      LET      x= 30
30      LET      y= x↑3 + 7* x
35      PRINT    y
40      LET      x= 50
45      LET      y= x↑3 + 7* x
...      ...      ...

      END
  
```

أي ، يجب وفي كل مرة نريد حساب قيمة y ، أن نعطي المتحولة x قيمة ثابتة ، ونكتب البرنامج الذي يحسب الدالة y . لذلك ، يجب كتابة البرنامج لمرات عديدة تعادل عدد القيم المعطاة للمتحولة x . وهذا لتفادي عملية التكرار هذه ، نكتب البرنامج على الشكل التالي باستعمال GOTO :

```

10      READ      x
15      LET      y= x↑3 + 7* x
  
```

```

20  PRINT  y
25  GOTO  10
30  DATA  20, 30, 50, 60, ...
35  END

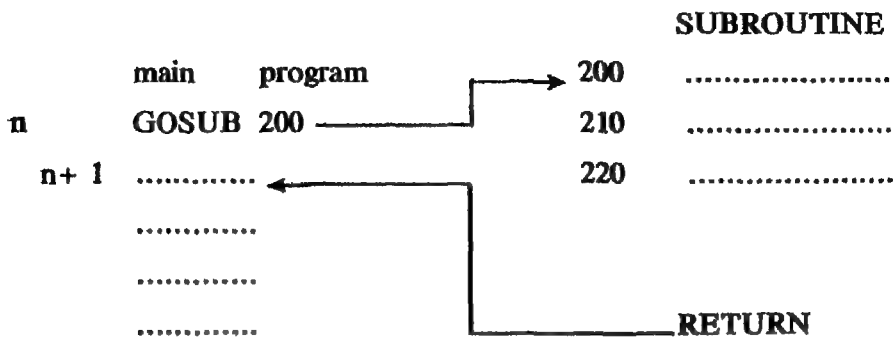
```

يعمل البرنامج بالطريقة التالية :

في البداية وبواسطة الأمر والتعليمات READ و DATA ، تقرأ الآلة قيمة المتحولة x وتحسب قيمة الدالة y . وبعد ذلك وبواسطة التعليمة ذات الرقم 25 (GOTO 25) تُحوّل مسار البرنامج إلى التعليمة رقم 10 ، فتعيد قراءة القيمة الثابتة للمتحوّلة x وتحسب الدالة y . تتابع الآلة هذه العملية حتى تنتهي من جميع القيم الثابتة المُعطاة للمتحوّلة x، والمذكورة بعد الأمر DATA . وهكذا نكون قد تفادينا كتابة البرنامج لمرات عديدة .

11.2 - الأوامر المستعملة لدعوة البرامج الثانوية SUBROUTINE

GO SUB- RETURN



يُستعمل الأمر GOSUB للدخول إلى البرامج الثانوية ، ويجب أن يكون متبوعاً برقم الأمر أو التعليمة الذي منه سندخل إلى البرنامج الثانوي الموجود في الذاكرة مع البرنامج الرئيسي .

K GOSUB n

n - رقم الأمر أو التعليمة الأولى للبرنامج الثانوي (SUBROUTINE) ، التي

منها سندخل فيه .

يتمهي البرنامج الثانوي بالامر RETURN ، الذي يعيد العمل إلى البرنامج المركزي من النقطة أو المكان الذي إنطلقنا منه ، أي من التعليمة التي تتبع رقم الامر GOSUB أي من التعليمة رقم $n + 1$.
مثلاً :

```
100 INPUT " LEBANON " ; A $
110 INPUT "BEIRUT " ; B $
120 INPUT " JADDA " ; D $
130 RETURN
```

وبالإمكان دعوة هذه المتحولات إلى البرنامج المركزي بواسطة :

```
10 ...
20 GOSUB 100
30 ...
```

11.3 - الأمر المشروط GOTO

الشكل العام لهذا الأمر هو :

$ON < expression > GOTO < n_1, n_2, \dots, n_k >$

في البداية تقوم الآلة بحسابة قيمة المعادلة expression وتأخذ منها القيمة الصحيحة ، فإذا كانت القيمة :

$expression = 1$ ، يذهب البرنامج إلى التعليمة ذات الرقم n_1 وذلك حسب

التسلسل التالي :

$expression = 1 \rightarrow GOTO n_1$

$expression = 2 \rightarrow GOTO n_2$

$expression = k \rightarrow GOTO n_k$

مثلاً :

```
A = 2
B = 3
ON ((A + B) / 2) GOTO 100, 110, 220, 230
```

$$\left| \frac{A+B}{2} \right| = 2 \quad \text{فإن} \quad \frac{A+B}{2} = 2.5 \quad \text{وبما أن :}$$

فسيدُهب البرنامج إلى التعليمة ذات الرقم 110 (GOTO 110) ، أي التعليمة الثانية في لائحة أرقام التعليمات .

إذا كانت قيمة المعادلة تزيد على 255 ، أو أن الأمر (التعليمة) ذو الرقم المطلوب غير موجود في لائحة أرقام الأوامر ، فسُتُعطي الآلة إشارة إلى حدوث خطأ مثلاً : ILLEGAL FUNCTION LINE XXX (سطر غير مُحدد) .

الأمر المشروط ON-GOSUB

مثلاً :

20 ON A GOSUB 300, 400, 515, ..

25 ...

.. ...

300 ...

350 RETURN

400 ...

.. ...

515 ...

.. ...

700 RETURN

.. ...

هذا البرنامج معناه :

إذا كانت $A = 1$ يتم تنفيذ التعليمات من الرقم 300 حتى 350
 $A = 2$ يتم تنفيذ التعليمات من الرقم 400 حتى 460 .

A = 3 يتم تنفيذ التعليمات من الرقم 515 حتى 700 .

....

الخ

11.4 - الأمر المشروط IF

الشكل العام لهذا الأمر هو :

IF (relative expression) GOTO LN 1

IF (relative expression) THEN LN 1 ELSE LN 2

LN 1 ، LN 2 ، عبارة عن أرقام الأوامر التي سيتم تحويل مسار البرنامج إليها حسب نتيجة الشرط المذكور في العبارة relative expression .

relative expression (يمكن أن تكون عبارة عن متحولة فقط) - معادلة شرطية ، منطقية أو حسابية ، ولها قيمتان واحد (1) ويعني أن الشرط المذكور فيها هو نافذاً. وصفر (0) ويعني أن الشرط ليس نافذاً وعندها يتم تنفيذ الأمر LN 2 .

بدلاً من استعمال أرقام الأوامر LN 1 ، LN 2 ، يمكننا أن نذكر بشكل مباشر العملية المطلوب إجرائها (أو التعليمة) في حالة تنفيذ الشرط أو عدم تنفيذه .

أي : IF (relative expression) THEN (operator) ELSE (operator).
أمثلة :

```
IF      A= B      GOTO      i00
IF      A= B      THEN      100
IF      A= B      THEN      100 ELSE 110
IF      A= B      THEN      M= "BEIRUT"
IF      A= B      THEN      C= 7 * 3 - D * 4 + 7
IF      A= B      THEN      C= 7 * 3 ELSE 100
IF      A= B      THEN      C=  LEBANON
                        ELSE      C=  BEIRUT
```

الشروط المستعملة في المعادلة المشروطة هي :

> أكبر
<> لا يعادل

أصغر
 يعادل
 = أصغر أو يعادل
 = أكبر أو يعادل

كما ويمكن للمعادلة أن تحتوي على عمليات منطقية من نوع OR ، D ، NOT . مثلاً :

A = 1 OR B = C THEN D = "BEIRUT" ELSE 100
 (A > B - c) OR (D = 5 AND M = N) THEN 10 ELSE 100
D

أمثلة على إدخال أوامر جديدة في الأمر IF

= B THEN PRINT "BEIRUT" ELSE 10
 = B THEN PRINT "BEIRUT" ELSE PRINT "LEBANON"

11.5 - الأمر ON-ERROR

الشكل العام هو :

ON ERROR < LINE NUMBER >

هذا الأمر يجب أن يتبع الأمر الذي يُسبب الخطأ (الإشارة التي تعطى عن ذلك) .

LINE NUMBER - هو الأمر (أو التعليمة) المفروض تنفيذه في حدوث الخطأ . رقم الإشارة يتم تخزينه في المتحولة EN ، والأمر ذو الرقم : NUMBER ، يقوم بتحليل رسالة الآلة عن الخطأ ، ويقرر ما يمكن عمله
 مثلاً :

OPEN "DK 1: xxx5" ON #10 FOR UPDATE
 ON ERROR 200
 :::
 IF EN = 82 OR EN = 83 THEN PRINT " PLEASE
 AL CORRECT DISK AND PRESS RETURN":
 TA: GOTO 10

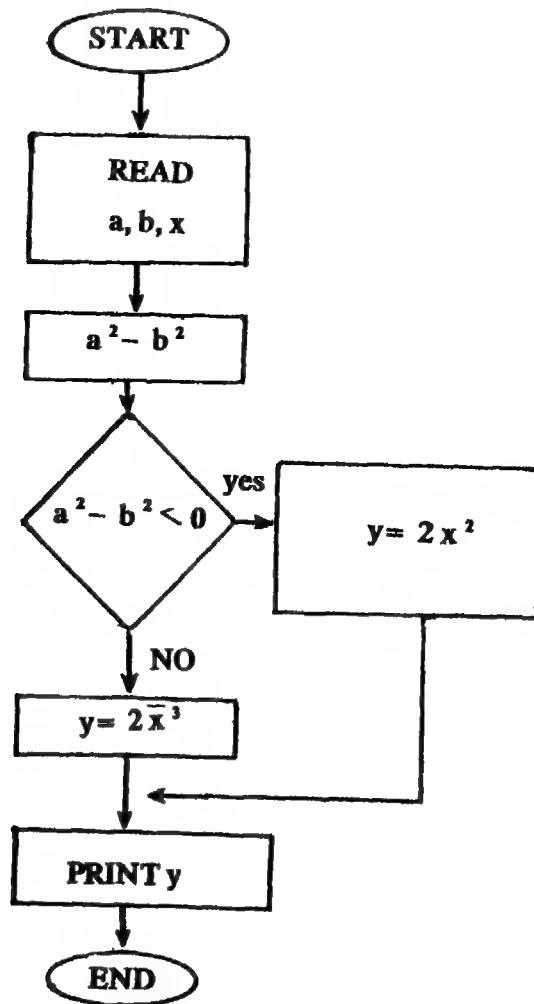
مسألة : أحسب قيمة الدالة y على الشكل التالي :

$$y = 2x^2 \text{ if } a^2 - 2b^2 < 0 \quad (1)$$

$$y = 2x^3 \text{ if } a^2 - b^2 \geq 0 \quad (2)$$

$$a = 28, b = 14, x = 12$$

البرنامج يجب أن يفحص قيمة المعادلة $a^2 - b^2$ ، فإذا كانت أصغر من « صفر » ، يقوم بحساب y حسب المعادلة الأولى $y = 2x^2$ وإلا فيحسب المعادلة الثانية $y = 2x^3$.



البرنامج :

```
100 REM TRANSFER CONTROL EXEMPLE
200 READ A, B, X
300 IF A↑2- 2* B 2 < 0 THEN 600
400 LET y= 2* x↑3
500 GOTO 800
600 LET y= 2* x↑2
700 PRINT y
800 DATA 28, 14, 12
900 END

RUN
```

...

مسألة : أكتب البرنامج الذي يحسب مجموعة مئة عدد من الجدول A .

```
10 REM PROGRAM FOR SUM OF 100 NUMBERS
20 DIM A (100)
30 S= 0
35 READ A (I)
40 S= S+ A (I)
50 I= I+ 1
60 IF I < 100 THEN 35
70 PRINT S
80 END
```

عملية الجمع هنا ، تتم على شكل حلقة (سلسلة من التعليمات)
(loop) ، أي في كل مرة نزيد فيها من قيمة المؤشر I (Index) الذي يدل على
الأعداد في الجدول A(100) . وعندما تصبح I معادلة لمئة (100) ، سيكون هذا
معناه ، أن عملية جمع المئة عدد قد انتهت ، فسنخرج بعد ذلك من جسم
الحلقة .

11.6 - الحلقات Looping والتعليمات FOR-TO-STEP-NEXT

الحلقات هي عبارة عن أقسام من البرنامج أو سلاسل من التعليمات يتم تنفيذها لمرات عديدة حسب قيمة متحولة عينة .

مثلاً : لنفترض أننا نريد أن نجمع مئة عدد من جدول معين فالبرنامج هو :

DIM A (100)

S = 0

S = S + A (1)

S = S + A (2)

... ..

... ..

S = S + A (100)

أي سنقوم بكتابة $S = S + A (I)$ مئة مرة في البرنامج . لذلك وبدلاً من هذا نستطيع تشكيل حلقة تقوم بعملية الجمع لمئة مرة باستعمال أحد الأوامر الإدارية من نوع IF- THEN أو الأمر FOR-TO .

الشكل العام لهذا الأمر هو : FOR V = n_1 TO n_2 STEP n_3 ;

... ..

NEXT V

V - متحولة . تستعمل كمؤشر (pointer) .

n_1 - القيمة الدنيا للمتحولة V .

n_2 - القيمة العليا للمتحولة V .

n - القيمة المضافة (ويمكن أن تكون سلبية) إلى المتحولة V في كل مرة يتم

تنفيذ الدائرة . أي أن V تتغير على الشكل التالي :

$$V = n_1 + n_3, n_1 + 2n_3, n_1 + 3n_3, \dots \leq n_2$$

وفي حالة غياب الجملة STEP n_3 فهذا معناه أن : $n_3 = 1$.

والآن لنكتب البرنامج الذي يقوم بجمع مئة عدد من الجدول A .

10 DIM A (100)

20 S = 0

```

30  FOR I= 1 TO 100
40  READ A (I)
50  S= S+ A (I)
60  NEXT I
70  DATA 1,2,3, ...
80  PRINT S
90  END

```

مسألة :

اكتب البرنامج الذي يجمع الأعداد المفردة من العدد 75 وحتى 101 .

```

50  SM= 0
60  FOR A= 75 TO 101 STEP 2
70  SM= SM+ A
80  NEXT A
90  PRINT SM
100 END

```

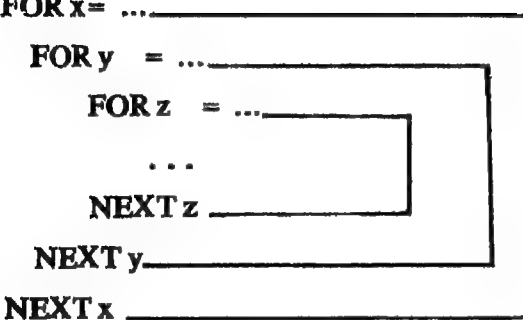
يمكن للدوائر أن تكون متداخلة الواحدة في الأخرى (NESTED LOOPS)

على الشكل التالي :

```

FOR x= ...
  FOR y = ...
    FOR z = ...
      ...
    NEXT z
  NEXT y
NEXT x

```



مسألة : معنا الجدول المربع A التالي :

(A)	1	2	3	4
1	0	5	6	7
2	8	9	10	11
3	0	0	1	2
4	3	5	7	8

اكتب البرنامج الذي يحسب مجموع أعداد الجدول (A) .
الحل : الجدول A هو مُربّع من نوع A (I, J) ، ويحتوي على أربعة أسطر وأربعة أعمدة . ولقراءته سنستعمل مؤشرين أحدهما يدلّ على الأعمدة والآخر على الأسطر . هنا سنقوم بتشكيل حلقتين متداخلتين الواحدة في الأخرى .

```

100   DIM   A (4, 4)
110   FOR   1 TO 4
120   FOR J= 1 TO 4
130   READ  A (I, J)
140   NEXT  K
150   NEXT I
160   REM   READING OF ARRAY A      قراءة الجدول A
170   FOR I= 1 TO 4
180   FOR J= 1 TO 4
190   S= S+ A (I, J)
200E   NEXT J
210   NEXT I
230   PRINT S
240   DATA 0 5 6 7 8 9 10 11 0 0 1 2 3 5 7 8
250   END

```

الحلقة الأولى تقرأ الأسطر ، والثانية تقرأ الأعمدة على الشكل التالي : في البداية تأخذ I القيمة واحد ($I = 1$) أي سيتم قراءة السطر الأول . بعد ذلك تبدأ المتحولة المؤشر J (pointer) بالتغيّر من 1 إلى 4 ($I = 1, 2, 3, 4$) ، كي تستطيع الآلة

أن تقرأ الأعداد الموجودة على السطر الأول والأعمدة الأربعة ذات الأرقام من واحد إلى أربعة أي : $A(1, 1), A(1, 2), A(1, 3), A(1, 4)$
وعندما تصبح $J = 4$ ، تقوم الآلة بتنفيذ التعليمة رقم 150 من البرنامج وهو
NEXT I . عند ذلك تصبح I معادلة لإثنين (أي $I = 2$) ، تقوم بعدها الآلة بقراءة
الأعداد الموجودة على السطر الثاني والأعمدة الأربعة ، أي :
 $A(2, 1), A(2, 2), A(2, 3), A(2, 4)$
وعملية القراءة من الأعمدة تتم بواسطة الأوامر :

FOR J= 1 TO 4

NEXT J

إلخ ، تتابع هذه العمليات حتى تصبح I معادلة لأربعة ($I = 4$) ، فنكون قد انتهينا
من عملية قراءة جميع الأعداد فنبدأ بعملية الجمع التي تجري بنفس الطريقة .

12 - البرامج الداخلية الثانوية SUBROUTINE

لتفادي تكرار بعض أقسام البرنامج ، ولتسهيل تصحيح وإختبار ومراقبة
عمل البرامج يتم تقسيمها إلى أقسام يؤلف كل منها برنامج داخلي بحد ذاته . كما
إن هناك العديد من البرامج الداخلية التي تقوم بحل العديد من المسائل الرياضية
والفيزيائية والاقتصادية والعلمية ، وموجودة في مكتبة البرامج ولا داعي لإعادة
كتابتها في البرنامج المركزي بل يجري دعوتها فقط .

جميع هذه البرامج تدعى البرامج الداخلية الثانوية وهي نوعان :

أ - البرامج الداخلية الثانوية من نوع دالة FUNCTION .

ب - البرامج الداخلية الثانوية من نوع منهاج ثانوي SUBROUTINE .

والآن لنستعرض هذه الأنواع من البرامج .

أ - البرامج من نوع دالة (FUNCTION) .

وهناك نوعان من هذه البرامج :

1 - برامج داخلية تباع مع الآلة وتدخل في صلب تصميم نظام التشغيل
الخاص باللغة المستعملة ؛ وتستعمل بشكل مباشر في البرامج كمتحولة من
المتحولات المستعملة في المعادلات . وأهم هذه البرامج هي تلك التي تقوم بحل
بعض المعادلات والدالات الرياضية شكل هذه البرامج هو :

FUN (arithmetical expression)

وأهم أنواع هذه البرامج المستعملة هي :

1- SIN < معادلة حسابية > ,

مثلاً :

$$\text{SIN}(x), \text{SIN}(2x + 4)$$

2- COS < معادلة حسابية >

مثلاً :

$$\text{COS}(x) = \dots$$

$$\text{COS}(2 \cdot x + 3) = \dots$$

3- TAN < معادلة حسابية >

مثلاً :

$$\text{TAN}(x) = \dots, \text{TANGENT}(2x + 3) = \dots$$

4- ASN

5- ATN , ATN(x) = ARC tangent (x) .

.

.

.

...

6- LOG , y = LOG(x) = log(x)

7- EXP , Y = EXP(x) = C^x

8- SQR , Y = SQR(4) = 2

9- ABS , Y = ABS(x) = x , ABS(-4) = 4

10- SGN, SGN(-4) = -1, SGN(4) = 1

$$\text{SGN}(0) = 0$$

ب - البرامج الثانوية « دالة » التي تعالج السمات .

والشكل العام لهذه البرامج هو :

FUNCTION (< Character string >, < expression >)

- Character string - متحولة رمزية أبجديدية .
 - expression - معادلة حسابية أو عدد صحيح .
 - FUNCTION - برنامج دالة من نوع function .
- وأهم هذه البرامج هو :

1- LEFT \$ (X \$, n)

X \$ - عبارة عن متحولة رمزية
n - عدد صحيح .
مثلاً : لو افترضنا إن :

X \$ = BEIRUT
A = LEFT (X \$, 3) = BEI

2- RIGHT (X \$, n)

مثلاً :

X \$ = BEIRUT
A = RIGHT (X \$, 2) = BE

3- MIDS (< STRING > , < n1 > , < L >)

STRING - عبارة عن متحولة رمزية .
n1 - رقم السمة التي سنبدأ منها بقراءة المتحولة .
L - عدد الرموز المقروءة .
مثلاً :

X \$ = LEBANON
n1 = 2
L = 3
A = MID \$ (X \$, 2, 3) = EBA

4- MID \$ (< STRING > , < n1 > , < L >) = < STRING >

مثلاً :

X \$ = LEBANON

$$n_1 = 7$$

$$L = 5$$

$$X \$ = \text{MID } \$ (X \$, 7, 5) = 10452$$

المتحولة \$X\$ تصبح :

$$X \$ = \text{LEBANON } 10452$$

$$5- \text{LEN} (< \text{STRING} >) = N$$

N - ستصبح معادلة لعدد السمات الموجودة في المتحولة STRING .

مثلاً :

$$X \$ = \text{LEBANON}$$

$$N = \text{LEN} ("LEBANON") = 7$$

$$6- \text{VAL} (< \text{STRING} >)$$

VAL - عبارة عن دالة عددية ، تعادل قيمتها القيمة العددية لسلسلة السمات STRING الحسابية .

STRING - سلسلة من السمات أو الرموز تحتوي على رموز للأرقام 0 ÷ 9 ، والإشارات + ، - ، . ، E .

مثلاً :

$$A \$ = "21.7"$$

$$N = \text{VAL} (A \$) = 21.7$$

$$7- \text{NUM } \$ (< \text{numeric expression} >)$$

الدالة NUM تُحوّل العدد numeric expression إلى رموز وسمات .

مثلاً :

$$P = \text{NUM } \$ (10452.00) = "10452.00"$$

8- ASCII (< STRING >)

ASCII - عبارة عن عدد يعادل الكود العشري للرمز الأول في المتحولة
STRING
مثلاً :

A = ASCII ("MAURIS") = 77

الكود العشري للرمز M هو 77 .

9- CHR \$ (NUMBER)

CHR - عبارة عن متحولة رمزية (رمز) بكود ASCII يعادل NUMBER
مثلاً :

P \$ = CHR \$ (82) = "R"

لأن الكود ASCII للرمز R هو 82 .

10- INSTR (< STRING 1 >, < STRING 2 > { ,

< first symbol N > { , < L > })

STRING 2, STRING 1 - عبارة عن متحويلات رمزية .

first Symbol N - رقم الرمز الأول الذي منه سنبدأ البحث في STRING

1 عن المتحولة الرمزية STRING 2 .

INSTR - يعادل رقم مكان الرمز الأول في STRING 1 ، لوجود المتحولة
STRING 2

مثلاً :

A = INSTR ("LEBANON", "B") = 3

INSTR ("ABC ABC ABC", "B", 3) = 5

11- EXCHANGE

يقوم بتبديل متحولتين علديتين ، أو رمزيتين ، الواحدة بالأخرى ، عن طريق
تبديل عناوينهم دون التغيير أو التبديل في مضمون الخلايا .

EXCHANGE <V₁>, <V₂>

<STRING 1>, <STRING 2>

V₁, V₂ - متحولات عددية .

STRING 1, 2 - متحولات رمزية .

مثلاً : هذا البرنامج يقوم بإخراج مضمون الجدول A من المتحولات الرمزية .

10 SIZE= 100; DIM A \$(SIZE)

20 FOR J= 1 TO SIZE

30 FOR I= J TO 1 STEP- 1

40 IF A\$(I) < A\$(I-1) THEN EXCHANGE A\$(I),

A\$(I- 1)... ELSE I= - 1

50 NEXT I

60 NEXT J

مسألة : اكتب البرنامج الذي يطبع أحرف المتحولة الرمزية :

AS= LEBANON

10 A\$= "LEBANON"

20 W= LEN (A\$)

30 FOR I = 1 TO W

40 PRINT LEFT \$(A\$, I)

50 NEXT I

60 END

RUN

L

LE

LEB

LEBA

LEBAN

LEBANON

نفس النتيجة سنحصل عليها من البرنامج التالي :

```
10 A$= "LEBANON"
20 FOR I= 1 TO LEN ( A$ )
30 PRINT LEFT$ ( A$ , I)
40 NEXT I
```

RUN

L

LE

LEB

LEBA

LEBAN

LEBANO

LEBANON

12- PEEK

PEEK (adress)

بواسطة هذه الدالة ، سنحصل على مضمون الخلية ذات العنوان adress بالكود العشري .

13- POKE

POKE (adress, expression)

POKE ، تُخزن في الذاكرة ، قيمة المعادلة expression بالنظام الثنائي وذلك على العنوان adress .

14- TRACE

بواسطة TRACE ، نحصل على رقم كل أمر أو تعليمة من البرنامج أثناء تنفيذه ، على الشاشة الكاتودية للكمبيوتر .
NOTRACE ، تلغي عمل TRACE .

15- CONT

إذا كان تنفيذ البرنامج متوقفاً بسبب الأوامر STOP ، أو END ... فالأمر CONT - يأمر الآلة بمتابعة تنفيذ البرنامج .

16- RESET

يوقف العمل في تنفيذ الأوامر مباشرة .

17- LOAD- SAVE

LOAD ، يقوم بإدخال البرنامج ، المسجل على شريط ممغنط أو كاسيتة ، إلى الذاكرة المركزية للكمبيوتر . أما SAVE فيقوم بإخراج البرنامج وتسجيله على الكاسيتة أو على الأسطوانة .

18- PRECISION

الأمر PRINT يطبع الأعداد بكامل دقتها ، أي بطول 12 رمزاً رقمياً للعدد . وإذا زادت الأرقام في العدد عن 12 رقماً ، ستقوم الآلة عند ذلك بطباعة العدد بفاصلة متحركة .

الأمر PRECISION يراقب عملية إخراج الأعداد ودقتها . وشكله هو :

PRECISION (< expression >)

expression ، عبارة عن معادلة حسابية ، أو عدد بين $0 \div 11$. والأمر (< expression >) PRECISION ، يُصرِّح عن عدد (مواقع) الأرقام المستعملة في تشكيل العدد المخرج . ولو افترضنا ، أنه وبسبب حجم العدد لا نستطيع إخراجته بالدقة المصرَّح عنها ، فعندئذٍ ستقوم الآلة بإخراجه بفاصلة متحركة ، ويكون العدد expression معادلاً لعدد الأرقام في الجزء العشري mantisse .
مثلاً :

10 PRECISION

20 FOR I = - 5 TO 10

30 PRINT #PRINTER, 2* 10** I / 3

40 NEXT I

50 END

النتائج هي :

6.66667E- 06	66.6667	6.66667E+ 06
6.66667E- 05	666.667	6.66667E+ 07
6.66667E- 04	6666.67	6.66667E+ 08
6.66667E- 03	66666.7	6.66667E+ 09
6.66667E- 02	666666.7	
6.66667E- 01		
6.66667		

جـ- البرامج الداخلية الثانوية دالة FUNCTION المصممة من المبرمج .

كما ذكرنا فإن البرامج الثانوية FUNCTION ، تحصل على قيمة ثابتة ، تُعطى لإسم البرنامج الثانوي عند تنفيذها أو عند استدعائها من قبل المبرمج إلى البرنامج المركزي .

وبالإضافة للبرامج الثانوية FUNCTION ، الموجودة مع الآلة (SIN ، COS ، ...) ، فباستطاعة المبرمج أن يضع ويُصمّم هذه البرامج وأن يُعيد إستعمالها في برنامج المركزي حيث تدعو الحاجة لذلك .

والتصريح عن البرامج الثانوية المصممة من قبل المبرمج يتم بطريقتين ، على الشكل التالي :

a) DEF FN (Function name) (< List of variables >) =
... < expression >

مثلاً : برنامج لحساب مساحة الدائرة :

DEF FN SUR (R) = 3.14 * R * R / 2

function name - إسم البرنامج الثانوي ، ويختاره المبرمج (في هذه الحالة هو SUR) .

List of variables - لائحة بالمتحولات المستعملة في البرنامج الثانوي ، أي في المعادلة expression التي تُشكل جسم الدالة أو سلسلة التعليمات التي تُؤلف البرنامج دالة . وقيمة هذه المتحولات الثابتة تُأخذ من الجملة - التعليمية التي بواسطتها يتم إستدعاء البرنامج دالة function ، أي الأمر (Call function) .

(في هذه الحالة هي R) . وتُسمى هذه المتحولات بالمتحولات الشكلية أو الوهمية (Formal variables) .

أما المتحولات الموجودة في الجملة - التعليمية و التي بواسطتها يتم إستدعاء البرنامج الثانوي فتُسمى بالمتحولات الحقيقية أو الفعلية (actual variables) وبنتيجة تنفيذ البرنامج FN سنحصل على قيمة ثابتة هي في الواقع قيمة المعادلة . expression

expression - عبارة عن معادلة حسابية أو رياضية ، تُشكل جسم الدالة وتُستعمل المتحولات الشكلية المنصوص عليها في لائحة المتحولات (وهي في هذه الحالة $S(R) = \Pi R^2$) .

مثلاً : اكتب البرنامج الثانوي الذي يحسب قيمة الدالة :

$$HS = \frac{e^x - e^{-x}}{2}$$

```

10      DEF FN HS (X)= (EXP (x)- EXP (- x)) / 2
20      FOR Z= .01 TO .1 STEP .01
30      PRINT Z , FN HS
50      END
RUN
.01      .0100001 66667
.02      .02000 133336
.03      .0300004 500202
.04      .040010 667521
.
.
.
.1      .100166 75002

```

b) والشكل الآخر للبرنامج الدالة Function هو :

-DEFFN < function name > ((< variables list >))

FNEND { < expression > }
 FN RETURN { < expression > }

جسم البرنامج ، وهو عبارة عن سلسلة تعليمات بلغة « بازيك » ، ويجب أن يكون موجوداً بين الأوامر DEF FN و FNEND .

Function name - اسم البرنامج الثانوي ، وهو عبارة عن متحولة غير مؤشرة . وإذا كانت قيمة البرنامج function هي عبارة عن قيمة من الرموز أو السمات الأبجدية فعند ذلك يجب أن يكون اسم البرنامج متبوعاً بالرمز (\$) .

Variables list - لائحة بالمتحولات الشكلية ، المستعملة في جسم البرنامج والتي تتغير قيمتها باستمرار أثناء تنفيذه .

FNEND - تدل على نهاية البرنامج FN .

Expression - عبارة عن معادلات رياضية تؤلف بالحقيقة جسم الدالة function ، أو جسم البرنامج الثانوي .

وعند تنفيذ الأوامر FNEND و FN RETURN يتم إعادة البرنامج إلى النقطة التي تم استدعاء البرنامج فيها . ولدعوة البرنامج الثانوي function نستعمل الأمر : CALL .

وعند تنفيذ الأوامر FNEND و FN RETURN يتم إعادة البرنامج إلى النقطة التي تم استدعاء البرنامج الثانوي فيها .

ولدعوة البرنامج الثانوي function نستعمل الأمر أو التعليمة التالية :

CALL FN < name of function > (variables lists)

variables list - عبارة عن المتحولات الحقيقية التي بها سيتم حساب قيمة البرنامج - الدالة عند إستعماله في البرنامج المركزي .

يُستعمل الأمر CALL في حالة وجود البرنامج الثانوي في مكتبة البرامج المسجلة على أسطوانة أو شريط مغناطيسي . وفي حالة عدم وجود هذا البرنامج على الأسطوانة أو الشريط ، ستقوم الآلة بالإشارة لذلك بواسطة الرسالة التالية :

UNDEFINED USER CALL AT LINE xxx

(أمر CALL غير مُحدّد في السطر xxx) .

ويمكننا استعمال البرامج الثانوية بشكل مباشر إذا كانت صغيرة على الشكل
التالي :
$$\text{DEF FN A (x) = SQR (x} \uparrow 4)$$

وفي البرنامج المركزي سندعو هذا البرنامج لحساب قيمة الدالة (3) A .

20 LET R = FNA (3) + 10

المتحولة الشكلية x في البرنامج الثانوي ، ستأخذ القيمة (3) ، عند دعوة
البرنامج FNA (x) إلى البرنامج المركزي .

- لمحو برنامج ثانوي من نوع function نستعمل الأمر FRESH متبوعاً
باسم البرنامج :

FRESH FN (< function name >)

مسألة : أكتب البرنامج الذي يحسب قيمة الدالة :

$$PA = N! = 1.2.3.4....$$

```
10 DEF FN PA (N)
20 IF N= 0 THEN FN RETURN 1
30 PR= 1
40 FOR A= 1 TO N
50 PR= PR* A
60 NEXT A
70 FNEND PR
```

لحساب قيمة الدالة (3) PA سنكتب :

$$y = \text{FN PA (30)}$$

ويمكن كتابة البرنامج على الشكل التالي باستعمال FNEND

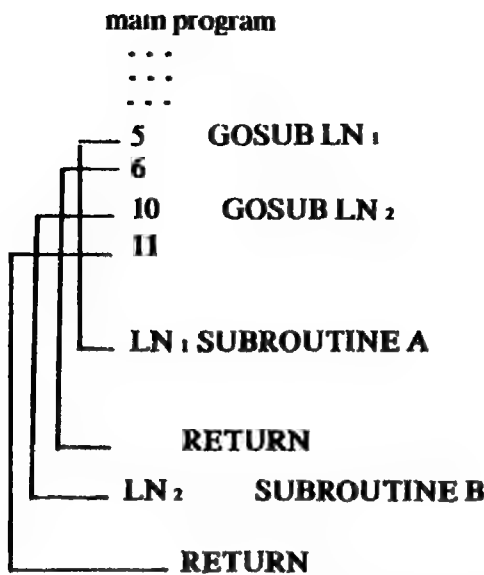
```
20 DEF FN PA (N)
30 IF N= 0 THEN FN RETURN 1
40 FNEND N FN PA (N- 1)
```

د- البرامج الثانوية من نوع SUBROUTINE

هذا النوع من البرامج ، يوضع من قبل المبرمج ، في الحالات التالية :

أ- عندما يكون جسم البرنامج المركزي كبيراً بشكل يصعب على المبرمج مراقبته وتصميمه وإختباره ، فيقسمه إلى أجزاء صغيرة يتم تصحيحها وإختبارها (TEST) ، وإستدعائها بعد التأكد من صحة عملها .

ب- عند وجود أقسام من البرنامج تتكرر لأكثر من مرة في البرنامج المركزي . عند ذلك سيكون من الأفضل كتابتها في نهاية البرنامج وإستدعائها عند الحاجة لذلك .



يتم استدعاء البرنامج Subroutine بواسطة الأمر .

GO SUB LN

LN - رقم الأمر الأول في البرنامج الثانوي subroutin .

ينتهي كل برنامج ثانوي بالأمر :

RETURN

الذي يعيد تنفيذ الأوامر إلى البرنامج المركزي في النقطة التي انطلق منها
للدعوته .

الفرق بين البرنامج SUBROUTINE و البرنامج FUNCTION ، أن
الثاني يعيد إلى البرنامج المركزي قيمة ثابتة ناتجة عن تنفيذه ، أما الأول فيعيد إلى
البرنامج المركزي قسم من البرنامج أو سلسلة تعليمات يتم تنفيذها فيه بعد إعطاء
المتحولات قيم ثابتة مُستعملة في البرنامج المركزي .

الفصل الثالث

الأوامر المتعلقة بتنظيم وإدارة الأجهزة والسجلات

13 - الأجهزة السجلات FILE and DEVICE

الأجهزة الفيزيائية للآلة الحاسبة هي :

- شاشة العرض التلفزيونية (display) CRT .
- لوحة ملامس الحاسب (KEYBOARD) ، KBD .
- آلة الطباعة (PRINTER) ، KBD .
- آلة الطباعة (PRINTER) ، PR .
- قنال الإدخال والإخراج (Serial in put/outout) ، SIO .

تعمل الأجهزة الفيزيائية للآلة الحاسبة بواسطة برنامج خاص تابع لنظام التشغيل ويدعى برنامج قيادة Drive program - يُخزّن هذا البرنامج تلقائياً في الذاكرة المركزية عند تشغيل الآلة الحاسبة .

الأجهزة المنطقية عبارة عن أسماء خاصة بالأجهزة تفهمها الآلة ، لهذه الأجهزة أرقام خاصة تُمنح للأجهزة الفيزيائية بواسطة الأوامر ASSIGN و OPEN ، لتصبح هذه الأخيرة جاهزة للعمل .

هناك نوعان من الأجهزة المنطقية :

- أجهزة التنظيم وهي :

أ- CONOUT # - شاشة الإخراج التلفزيونية (Console) ، تُعطي للجهاز CRT (Display, terminal) .

ب- CONTN # - لوحة ملامس الحاسب ، ويُعطى للجهاز الفيزيائي KBD .

ج- PRNTR # - للطباعة ، يعطى للجهاز الفيزيائي PR .

د- RDR # - جهاز للإدخال بذاكرة كبيرة (قارئ البطاقات المثقبة) .

هـ- PUNCH # - جهاز للإخراج بذاكرة كبيرة (مُثَقِّب البطاقات) .

و- AUX # - أجهزة منطقية إضافية .

- أرقام السجلات المنطقية .

عبارة عن أعداد لا تزيد عن 254 مسبوقة بالرمز # ، ولها الشكل التالي :

سجل ، فايل ، ملف : File

< expression >

مثلاً : # 10 ، # 105 ، # 2+ 3 ، # A+ B

12.1 - الأوامر الخاصة بالأجهزة: RELEASE ، ASSIGN

بواسطة الأوامر ASSIGN وRELEASE يتم تعليق أو إطلاق حرية أحد الأجهزة الفيزيائية بالأجهزة (من الأجهزة) المنطقية .

ASSIGN | < NO connected Logic file number > | To
| < Connected logic file number > | ,

| < system Logic device > | "To
| < system logic device > | ,

| < system Logic device > | TO
| < system physic device > |

RELEASE < Connected Logic file number >

(#0 ≤ connected logic file number ≤ # 254

	# CONOUT
system logic device =	# CONIN
	# PRNTR
	# RDR
	# PUNCH
	# AUX
	# CRT
system physique device =	KBD
	ECHO
	PR 1
SIO	SIO

أمثلة :

ASSIGN # CONOUT TO "PR 1"

أ -

يقوم بإخراج المعلومات المرسلة إلى الشاشة على آلة الطباعة PR 1 .

ASSIGN CONOUT TO "SIO"

ب -

يقوم بإخراج المعلومات المرسلة إلى الشاشة ، على القنال SIO .

10 A \$= "CONSOLE OUTPUT MESSAGE"

ج -

20 ASSIGN # CONOUT TO "PR 1"

30 PRINT A\$

سيتم إرسال المتحولة الرمزية A \$ إلى الطباعة PR 1

40 CLOSE # CONOUT

الجهاز # CONOUT سيعاد إلى شاشة العرض CRT display

50 PRINT A \$

المتحولة \$ A سيتم إرسالها إلى شاشة العرض CRT
ASSIGN PRNTR TO CRT

ستقوم الشاشة CRT بدور الطابعة PRNTR
RELEASE # PRNTR

فك ربط الجهاز المنطقي PRNTR # بالجهاز الفيزيائي CRT .

هـ - لنفترض أن السجل XREF هو عبارة عن سجل داخلي .
OPEN "DK 1: XREF" ON # 10 FOR UPDATE
يقوم بفتح الفايل XREF ويعطيه رقم منطقي هو 10 #

PUT # 10, A, B, C \$
يقوم بإرسال المعلومات A ، B ، C إلى السجل 10 # أي إلى السجل
الفيزيائي XREF .

GET 10, A, B, C \$

يقوم باستلام المعلومات من السجل XREF .

و - لنفترض إن 100 # عبارة عن رقم منطقي للسجل أو الفايل PR 1 .
ASSIGN # 100 TO PR 1

سيأخذ جهاز المطبعة المنطقي رقم السجل 100 # .
أي سيتم إرسال المعلومات من السجل 100 # إلى الطابعة PR 1 .
ASSIGN # 100 TO CRT

يتم إرسال المعلومات من السجل 100 # إلى شاشة العرض التلفزيونية .

14 - الأوامر والتعليمات المتعلقة بمعالجة الجداول والمصفوفات (MATRICE)

تحتوي لغة « البازيك » على العديد من الأوامر المتعلقة بمعالجة الجداول .
تبدأ هذه الأوامر بكلمة MAT وتنتهي بالمهمة المطلوبة من الأمر . أهم هذه الأوامر
هي :

MAT READ C	READ MATRIX	لقراءة الجدول أو المصفوفة
MAT PRINT C	PRINT MATRIX	اطبع المصفوفة
MAT C= TRN (A)	TRANSPOSE MATRIX	
MAT C= ZERO	ZERO MATRIX	مصفوفة أو جدول «صفر»
MAT C= IDN	IDENTIFY MATRIX	تعريف مصفوفة
MATC= CON	J- MATRIX	
MATC= A + B	ADD MATRIX	جمع مصفوفتين
MATC= A - B	SUBSTRACT MATRIX	طرح مصفوفتين
MATC= (A)* B	SCALAR MULTIPLICATION	ضرب مصفوفتين
	OF MATRIX	
MATC= A*B	MULTIPLY MATRIX	ضرب المصفوفات
MATC= INV (A)	INVERT MATRIX	قسمة المصفوفات

أمثلة :

10 DIM A (10, 12), C (30, 40)

هذا الأمر يحفظ للمصفوفة المسماة A ، عدداً يعادل (10 × 12) خلية ،
وللمصفوفة C عدداً يساوي (30 × 40) خلية من الذاكرة .

لقراءة المعلومات من الأمر DATA وإعطائها للمصفوفة نكتب :

MATREAD V1

DATA a1, a2, a3

مثلاً :

10 DIM A (4, 3)

20 MATREAD A

30 DATA 6, 7, 8, 9, 10, 17, 12, 15, 16, 17, 18

بواسطة هذه الأوامر، نُصرِّح عن المصفوفة A، ومن ثم نقرأها بواسطة الأمر
التالي 20 . وبعد ذلك نُخزِّن في خلاياه. المعطيات الواردة في الأمر DATA على
الشكل التالي :

$$A = \begin{vmatrix} 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 15 & 16 \\ 17 & 18 & 19 \end{vmatrix}$$

مثل : إطببع عشرة اعداد من المصفوفة A المؤلف من مئة عدد .

```

10  DIM A (100)
20  READ A
30  FOR I= 1 TO K
40  READ A (I)
50  NEXT I
60  DATA 10
70  DATA 8, 9, 4, 3, 10, 15, 38, 9, ...

```

وبالإمكان هنا أن نستعمل الأمر الخاص بطباعة المصفوفة .

MAT PRINT v

v - اسم المصفوفة المطلوب طباعتها .
مثلاً :

```

10  DIM A (2, 5)
20  MAT READ A
30  MAT PRINT A
40  DATA 10, 50, 60, 80, 90, 4, 3, 2, 5, 6
50  END

```

RUN

```

10    50    60    80    90
4     3     2     5     6

```

15 - الأوامر المستعملة للعمل مع الأسطوانات المغناطيسية وتنظيم العمل بالسجلات .

يجري تسمية الإسطوانات المغناطيسية (disk) ، والسجلات (File) والبرامج المستعملة بأسماء خاصة كتلك التي تُسمى بها المتحولات ، مع الفارق بأن طولها لا يتعدى 11 رمزاً . مثلاً . . . و

MASTER DISK ، PAY ROLL 1 ، PRINT FILE ، . . .

وبالإضافة للأسماء ، فللأسطوانات والسجلات أرقاماً خاصة بها وتعرف عليها (identifier number, ID) . يتم تسجيل رقم الأسطوانة ID في الذاكرة ، وتقوم الآلة بمقارنة هذا الرقم مع كل رقم إسطوانة أثناء عملية الإخراج والإدخال للتأكد من وجود الأسطوانة المطلوبة .

أما رقم الفاييل أو السجل (file identifier, ID) فيحتوي على معلومات خاصة حول الجهاز الموجود عليه ، إسم السجل ، نوع السجل ، إسم الأسطوانة . . . الخ :

ID = { | < DISK device > | , } < FILE name > { , < File TYPE
} | < DISK name > |

disque device - هو عادة إسم محرك جهاز الأسطوانات أو القارئ (Disk Drive) ، مثلاً : DK 0 ، DK 1 ، DK 2 ، DK 3 .

name DISK - إسم الأسطوانة ، المعمول بها بعد تنفيذ الأمر
FREBLOCK .

File name - إسم الفاييل أو السجل .

File TYPE - نوع الفاييل أو السجل ، وهو عبارة عن رقماً خاصاً يتغير من 0 إلى 255 . ولكل رقم في هذه الأرقام معنى خاصاً (يختلف حسب نوع الآلة ولذلك لن أدخل بالتفاصيل) . مثلاً : 250 TYPE معناه TEXT FILE أي سجل من النصوص .

مثلاً : «FILEA» - يصرّح عن فاييل أو سجل باسم FILEA .

«DKO : FILEB» - يصرّح عن فاييل أو سجلاً باسم FILEB ، موجوداً على

أسطوانة المدعوة DK 0 .

«BEIRUT: FILEC» - يصرّح عن سجلاً باسم FILEC - يصرّح عن سجلاً باسم FILEC ، موجوداً على اسطوانة باسم BEIRUT .

15 - التصريح عن الفايل والسجلات .

الامر DE FINE وشكله هو :

DEFINE < ID FILE > , RECORD SIZE= < integer > ,

NBR ... RECORDS= < integer 2 > { , INT }

العدد integer 1 - هو عدد البايتات في كل تسجيلية (record) أي طول

التسجيلية من السجل .

(1 < integer 1 < 4096)

integer 2 - يدل على عدد التسجيلات (records) في كل سجل .

(1 < integer 2 < 65, 535)

INT - غير إلزامية ، وتدل على إن السجل هو داخلي ومباشر ، وإلا (في حالة عدم ذكرها) فسيكون السجل للإخراج على المطبعة (PRINTING FILE) . وعند تنفيذ الأمر DEFINE تقوم الآلة بالبحث في الأسطوانات المربوطة بالآلة عن السجل المطلوب .

أما في الحالة التي تصرّح فيها ID عن :

أ - جهاز من أجهزة الأسطوانات (مثلاً ، DK1 ، « DK) ، فستقوم الآلة بتسجيل السجل على الجهاز المصرّح عنه .

ب - إسم الأسطوانة ، فستقوم الآلة بالبحث عنها في الأجهزة المعلقة بها بواسطة إسمها وعند إيجادها تقوم بتسجيل السجل على تلك الأسطوانة .

ج - أما إذا كانت ID لا تصرّح عن جهاز ولا عن أسطوانة ، فستقوم الآلة بتسجيل السجل على أحد الأجهزة التي تعمل معها ومربوطة فيها .

أمثلة :

DEFINE «DK»: RATES» , RECORD SIZE= 100 ,

NBR RECORDS= 500, INT

هذا الأمر يصرّح عن سجل مباشر (داخلي) ، للأسطوانة ذات الرقم 0 ،

واسم الفايل هو RATES ، ويحتوي على 500 تسجيلة بطول 100 بايتة للتسجيلة

```
DEFINE ``BEIRUT: LENON» , RECORD IZE= 10 ,  
NBR RECORDS= 100
```

هذا الأمر يُصرَّح عن سجل للطباعة بطول 1000 بايتة ، موجود على
أسطوانة باسم BEIRUT ، ويُدعى LEBANON .
15.2 - الأوامر المتعلقة بالعمل ومعالجة السجلات .

1 - الأمر «إفتح» ، OPEN

قبل إستعمال السجل ، يجب أن يتم فتحه بواسطة الأمر OPEN
OPEN < ID File > ON < No connected logic file N > ...
For | INPUT |
| OUTPUT |
| UPDATE |

NO Connected logic file N - عبارة عن رقم السجل المنطقي ، الذي
سيُستعمل في عمليات الإدخال والإخراج ، هذا السجل يجب أن يكون غير مربوط
في السابق (أي غير مفتوح قبل ذلك بواسطة الأمر OPEN) .
الخصائص الثلاثة تدل :

INPUT - تدل على أن السجل هو للقراءة .
OUTPUT - تدل على إن السجل هو للتسجيل .
UPDATE - تدل على إن السجل هو للتسجيل والقراءة .

أمثلة :

```
OPEN ``MASTER`` ON # 10 FOR UP DATE
```

هذا الأمر ، يقوم بعملية البحث عن السجل باسم MASTER ، بنوع "O" ، في

كل الأسطوانات الموجودة مع الآلة . رقم السجل المنطقي هو 10 # ويستعمل للقراءة والكتابة .

OPEN " SUBDISK: SUBFILEA. 7 " ON # 45 FOR OUTPUT

السجل SUB FILE A هو بنوع 7 ، موجود على الأسطوانة SUBDISK ، ورقمه هو 45 ، ويُستعمل للإخراج OUTPUT .

2 - الأمر CLOSE .

يُستعمل لإغلاق السجل بعد فتحه ، ويجب عدم استعماله بعد الآن .

CLOSE | < Logic file number > | { , PROT ...

| < logic device name > |

{ ECTE } { } , ... }

logic file number - رقم السجل المفتوح قبل ذلك بواسطة OPEN .

logic device name - هو اسم الجهاز المنطقي ويكون عادة أحد هذه

الأجهزة : # CONOUT ، # CONIN ، # PRNTR ، # RDR ، # PUNCH ، # AUX .

في حالة إستعمال PROT (PROTECTED) ، الفايل سيكون السجل محمي ضد الكتابة بعد إغلاقه . ولنستطيع التسجيل والكتابة عليه بعد ذلك يجب أن نستعمل الأوامر : RENAME أو REDEFINE .

مثلاً :

CLOSE # 15 و PROTECTED

سيتم إغلاق السجل ذو الرقم 15 # ، وسنمنع عنه عملية التسجيل والكتابة .

3 - الأمر CLEAR

CLEAR - يقوم بعملية تنظيف (محو) لجميع المعلومات الموجودة على

السجل المباشر . يجب أن يكون هذا السجل مفتوحاً للإدخال والإخراج (OPEN

(UPDATEE) ... قبل تنفيذ العملية CLEAR .

وشكل هذا الأمر هو :

```
CLEAR < logic file connected number >  
< Name of logic device >
```

(logic file connected number, name of logic device) لها نفس المعنى

السابق في الأمر Close .

مثلاً :

```
OPEN "BEIRUT" ON # 10 FOR UPDATE  
CLEAR # 10  
CLOSE # 10
```

4 - الأمر PURGE

PURGE ، يقوم بعملية محو لسجل من المعلومات على الأسطوانة .

```
PURGE < no connected logic file number >  
< logic device name >
```

مثلاً :

```
OPEN "TEST PROG. 254 " ON # 5 FOR UPDATE  
PURGE # 5
```

هذا المثل ، يقوم بمحو البرنامج المسمى TEST PROG ، الموجود في السجل ذو الرقم 2549 والمفتوح سابقاً للإدخال والإخراج .

5 - الأمر RENAME

يُستعمل لإعادة تسمية الفائل . ولتغيير نوعيته ، ويجب أن يكون السجل مفتوحاً قبل الأمر RENAME .

<logic device connected number>
 <logic device name>
 ... To <ID File> { , PR0T { ECTE } }

logic device connected number - هو رقم الجهاز المنطقي المستعمل سابقاً
 أثناء تنفيذ الأمر OPEN .

logic device name أحد الأجهزة المستعملة : CONOUT # ،
 # CONIN ، # PRNTR ، # RDR ، # PUNCH ، # AUX ،
 ...
 أمثلة :

```
OPEN "PAY 7" ON # 0 FOR INPUT
RENAME # 0 TO "PAY"
CLOSE # 0
```

هذا البرنامج يقوم بتغيير اسم السجل من PAY 7 إلى PAY .

6 - الأمر RESIZE

لتغيير حجم السجل المستعمل :

```
RESIZE | < connected logic file number > | ...
      | < logic device Name > |
      To < integer > { RECORDS }
```

integer - تدل على الحجم الجديد (أو عدد التسجيلات) في السجل . وفي
 الحالة التي يكون فيها integer = 0 ، سيتم محو السجل .
 مثلاً :

```
OPEN "HENRY" ON # 20 FOR INPUT
RESIZE # 20 TO 2000 RECORDS
```

السجل HENRY سيكون حجمه 200 تسجيلة .

7 - الأمر COPY

يقوم بعملية إعادة كتابة أو نسخ المعلومات من أحد السجلات إلى آخر .

COPY < ID file 1 > TO < ID file 2 >

السجل 1 file سيتم نسخه في السجل 2 file .

8 - الأمر REDEFINE

يستعمل لتغيير بعض المعلومات المتعلقة بالسجل وأهمها :

- 1 - اسم السجل
- 2 - وضع السجل ونوعه (بحماية ضد التسجيل أو بدونه)
- 3 - عدد التسجيلات في السجل .
- 4 - معلومات حول السجل .

REDEFINE | < logic file connected number > |

| < logic device name > | .

.... { , NAME = < ID file > { , PROT = > < string 1 > }

.... { , NBR RECORDS = < integer 1 > }

{ , PW = < string 2 > }

مثلاً :

```
10 OPEN "BEIRUT" ON # 20 FOR INPUT,
    PW = "GOODBYE"
20 REDEFINE # 20 , NAME = "SAMPLE",
    NBR RECORDS = 400 .
    PW = "HELLO" , LOCK = "ON"
30 CLOSE # 20
```

9 - الأمر CLOSE ALL

يستعمل لإغلاق جميع السجلات المفتوحة في البرنامج بواسطة OPEN .

10 - الأمر FIND

يبحث عن التسجيلة المطلوبة (active record) ليصبح جاهزاً للعمل به بواسطة PUT أو GET للقراءة أو الكتابة .

```
FIND | < connected logic file number > | , | NEXT | | integer |  
| < logic device name > |
```

باستعمال NEXT سيتم إضافة واحد (1) إلى رقم التسجيلة بداخل السجل ، وذلك للحصول على جميع التسجيلات بشكل متتابع . (active record) .

باستعمال integer : التسجيلة المطلوبة (record active) ستحصل على رقم معادل للعدد integer ، على أن لا يزيد هذا الأخير عن عدد التسجيلات بداخل السجل .

```
OPEN "STOCK" ON # 7 FOR UPDATE
```

```
FIND # 7,21
```

التسجيلة المطلوبة ستحصل على الرقم 21 .

```
FIND # 7,25
```

التسجيلة المطلوبة ستحصل على الرقم 25 .

11 - الأمر PUT

هذا الأمر يقوم بتسجيل المعلومات في السجل المباشر ، ولا يمكن استعماله في السجل المتتابع (sequential file) (هناك يُستعمل فقط الأوامر PRINT USING , PRINT) .

```
PUT | < non connected logic file > | , < variables list > { , }  
| < name of logic device > |
```

إذا كان السجل غير مربوط (non connected logic file) ، فيجب فتحه قبل عملية التسجيل أو الكتابة عليه للإخراج (OUTPUT) ، أو للإدخال والإخراج (UPDATE) .

وإذا وضعنا فاصلة (,) في نهاية الأمر ، سيتم إخراج جميع المعلومات الموجودة في التسجيلية . وفي حالة عدم وجودها ستقوم الآلة بإخراج المعلومات حتى بداية التسجيلية التالية . (لذلك لا يجب وضع الفاصلة عند إخراج المعلومات إلى التسجيلية الأخيرة في السجل) . أما إذا تم وضعها فتقوم الآلة بالإشارة إلى حدوث خطأ ، حيث لا يوجد تسجيلية أخرى في السجل .
مثلاً :

```
10 OPEN "START" # 10 FOR UPDATE
20 FIND #10, 37
30 PUT #10, A, B, C $, D, E $
40 PUT #10, E, F, G $
```

الأوامر PUT في هذا المثل ستقوم بإخراج المعلومات من التسجيلية رقم 37 .

12 - الأمر GET

يقوم بقراءة المعلومات من السجل المباشر .

```
GET | < connected logic file number > | , < variables list > { , }
| < logic device name > |
```

الفاصلة لها نفس المعنى السابق .

Variables list ، عبارة عن لائحة بأسماء المتحولات المطلوب قرائتها ، ويمكن أن تكون عددية أو رمزية . وعندما تقوم الآلة بقراءة الرموز بواسطة GFT ، تقرأ أولاً طول المتحولة الرمزية المسجلة سابقاً وبشكل أوتوماتيكي بواسطة PUT

أمثلة :

```
10 OPEN BEIRUT "ON" # 10 FOR INPUT
20 GET # 10, A, B, C
```

الأمر 10 يفتح السجل المسمى BEIRUT للإدخال .

والأمر الثاني يقرأ المتحولات A, B, C من التسجيل رقم 10 للسجل
BEIRUT .

30 GET # 10, D, EF

الأعداد الثلاثة الأولى للتسجيلية رقم 2 من السجل BEIRUT تتم قراءتهم
في المتحولات ، D, E, F .

13 - الأمر DROP

يجعل المعلومات الموجودة في أحد تسجيلات السجل غير جاهزة للقراءة
والكتابة .

DROP | < connected logic file number > |
| < logic device name > |

مثلاً :

OPEN "BEIRUT" ON # 14 FOR OUTPUT
FIND # 14, 157
DROP # 14

التسجيلية رقم 157 من السجل المسمى BEIRUT سيصبح غير قابل
للتسجيل والقراءة .

14 - الأمر SAVE

الأمر SAVE يحفظ برنامجاً بلغة « بازيك » في سجل على الأسطوانات، أو على
الأسرطة المغناطيسية .

SAVE { < ID FILE > }

المعرف ID- Identifier ، يُصرّح عن الأسطوانة التي سيُحفظ عليها
البرنامج وعن اسمه . ويعتبر ID غير ضرورياً إلا في الحالات التي سيكون فيها
البرنامج مُخزناً في الذاكرة بواسطة الأوامر CHAIN, LOAD . في هذه الحالات
وإذا لم يُذكر المعرف ID ، ستقوم الآلة بواسطة SAVE بتغيير اسم البرنامج الموجود
على الأسطوانة بهذا الاسم الموجود في الذاكرة .

مثلة :

SAVE "PROGRAM 1"

البرنامج سيُحفظ على الأسطوانة تحت الاسم PROGRAM 1 .

SAVE "DK 0; PROG2"

البرنامج سيُحفظ على الأسطوانة الموجودة على الجهاز DK 0 ، باسم

PROG2 .

LOAD_ 15

يقوم بتخزين البرنامج المسجل على الأسطوانة في الذاكرة .

LOAD < ID FILE >

مثلاً : "PROGRAM 1"

البرنامج PROGRAM 1 سيُخزن في الذاكرة الداخلية للآلة .

LOAD "DK 0: PROG 1"

البرنامج PROG 1 ، الموجود على الأسطوانة DK 0 سيُخزن في الذاكرة .

CHAIN_ 16

يُستعمل الأمر CHAIN في أحد البرامج ، ليقوم بتخزين وتنفيذ برنامجاً آخر . المتحولات الموجودة في البرنامج السابق ستكون تحت طلب البرنامج الجديد كذا ، بالنسبة لجميع السجلات المفتوحة في البرنامج السابق .

CHAIN < ID file >

مثلاً :

CHAIN "PAY"

سيتم تخزين وتنفيذ البرنامج PAY

CHAIN "PROGS: PAY"

سيتم تخزين البرنامج PAY الموجود على الأسطوانة PROGS وتنفيذه .

MERGE_ 17

يأخذ أحد البرامج من الأسطوانة ويمزجها بالبرنامج المركزي الموجود في ذاكرة الآلة حسب ترتيب أرقام الأوامر والتعليمات .

MERGE < ID file >

MERGE < ID file > { , PW = < string exp > }

APPEND_ 18

يأخذ أحد البرامج الموجودة على الأسطوانة ويُعلقه في نهاية البرنامج الأصلي الموجود في ذاكرة الآلة .

APPEND < ID file >

ASAVE_ 19

يحفظ البرنامج الموجود في ذاكرة الآلة على اسطوانة بشكل جديد . (EXTENDING FORMAT)

ASAVE < ID file >

LOADA_ 20

يستعمل لتخزين برنامجاً بلغة الآلة .

LOADA < ID file >

PLUSH_ 21

لمحو أحد البرامج المسجلة في الذاكرة بلغة الآلة منها .

PLUSH < ID file >

ID - هو رقم مُعرّف للسجل .

CALLA_ 22

يُستعمل لدعوة وتنفيذ أحد البرامج المكتوبة بلغة الآلة .

CALLA < ID file > { , < list of variables sring > }

*** هذه هي أهم الأوامر المتعلقة بالعمل مع الأجهزة والفايل ، وقد كتبتها باختصار شديد لأنها غير ثابتة وتتعلق إلى حد كبير بتصميم الآلة وبرنامج التشغيل المتبع فيها ، ولذلك يجب على المبرمج أن يطلع على الوثائق التابعة للآلة والخاصة بنظام عمل الآلة بلغة « بازيك » .

الفصل الرابع
مسائل مبرجة بلغة

- BASIC -

مسألة 1 :

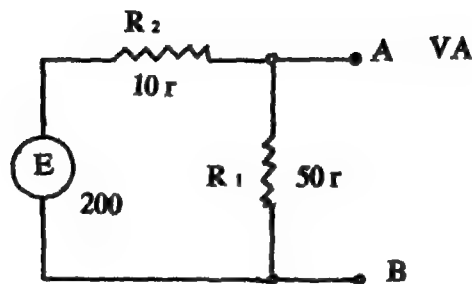
اكتب البرنامج الذي يقوم بحسابة وطباعة قيمة المتحولة R حسب المعادلة التالية .

$$R = \sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}$$

$$a_1 = 4, a_2 = 8, a_3 = 29, a_4 = 170$$

البرنامج :

```
100 REM SAMPL PROGRAM
110 DIM a(4)
120 r=0
130 FOR i=1 TO 4
140 READ a(i)
150 r=r+a(i)^2
160 NEXT i
170 r=SQR(r)
180 PRINT r
190 DATA 4,8,29,170
200 END
```



مسألة 2 :

احسب قيمة الفولطية V في النقطة A للدائرة الكهربائية الموجودة في الصورة أعلاه .

الحل :

الفولطية V_A في النقطة A يعادل :

$$V_A = \frac{E}{R_1 + R_2} \cdot R_1$$

البرنامج :

```
10 REM CIRCUIT ELECTRIC
20 READ e,r1,r2
30 v=e*r1/(r1+r2)
40 PRINT v
50 DATA 200,50,10
60 END
```

مسألة 3 :

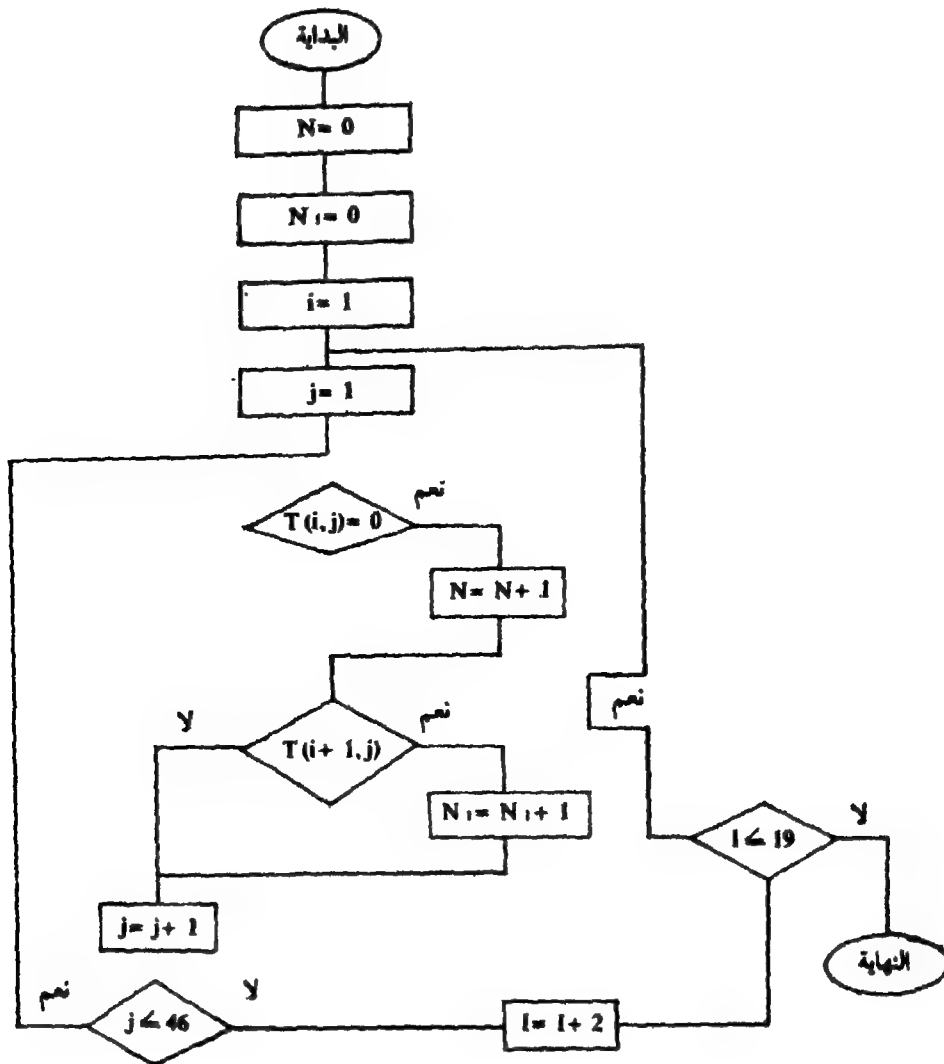
معنا جدول مربع هو $T(20,46)$. اكتب البرنامج الذي يقرأ هذا الجدول . ومن ثم يحسب عدد الخلايا (أو المواقع) الفارغة N ، والموجودة على الأسطر ذات الأرقام المفردة (1, 3, 5, ..., 19) . وعدد الخلايا الموجودة على الأسطر المزدوجة (2, 4, ..., 20) ، والتي تعادل بمضمونها (1 -) .

مثل :

$A(6, 6)$

	1	2	3	4	5	6
1	1	0	2	-1	-3	-4
2	-1	x	2	0	3	x
3	x	x	2	-1	x	x
4	5	6	4	-1	x	x
5	-1	3	x	x	-1	x
6	x	-1	x	x	-1	x

الأسطر ذات الأرقام المفردة هي : 1, 3, 5 ، عدد الخلايا الفارغة على هذه الأسطر هي : 7. عدد الخلايا ذات المضمون (-1) والموجودة على الأسطر المزدوجة هي : 4 .



PROGRAM MAIN

```

10 DIM t(20,46)
20 n=0
30 n1=0
40 FOR i=1 TO 19 STEP 2
50 FOR j=1 TO 46
60 READ t(i,j)
70 IF t(i,j)>0 THEN 90
80 n=n+1
90 IF t(i+1,j)<>-1 THEN 110
100 n1=n1+1
110 NEXT j
120 NEXT i
130 DATA
140 PRINT "n= ";n
150 PRINT "n1=";n1
160 END

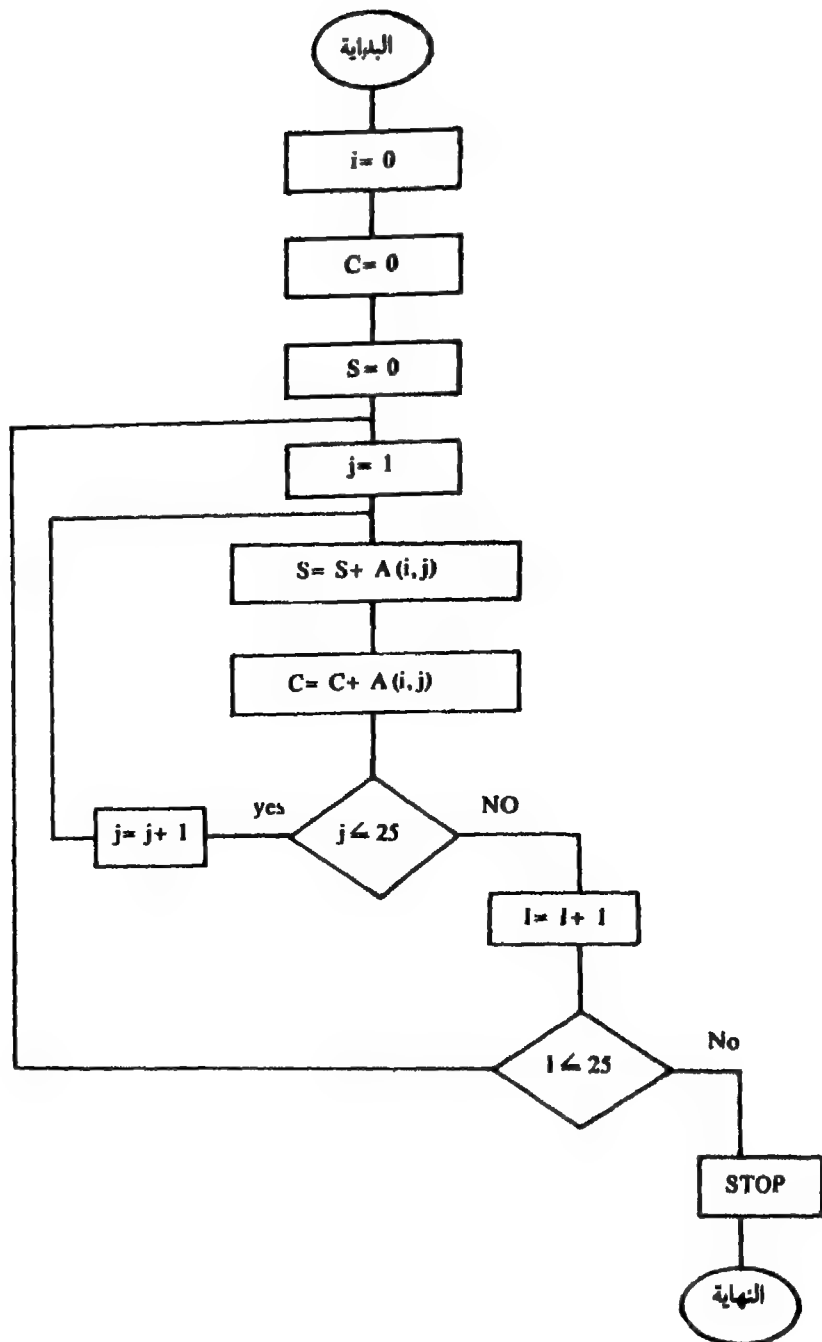
```

مسألة 4 :

معنا الجدول المربع التالي (25, 25) A
ضع البرنامج الذي يحسب :

- مجموع مضمون الخلايا في كل خط أفقي S .
- مجموع مضمون الخلايا في كل عمود C .

- 1 - عبارة عن مؤشر (Pointer) للأسطر .
- 2 - عبارة عن مؤشر (pointer) للأعمدة .



```

10 REM SUM PROGRAM MAIN
20 DIM a(25,25)
30 FOR i=1 TO 25
40 FOR j=1 TO 25
50 s=0
60 c=0
70 READ a(i,j)
90 q=s+a(i,j)
100 NEXT j
110 PRINT"s=";s
120 PRINT"c=";c
130 NEXT i
140 DATA
150 STOP
160 END

```

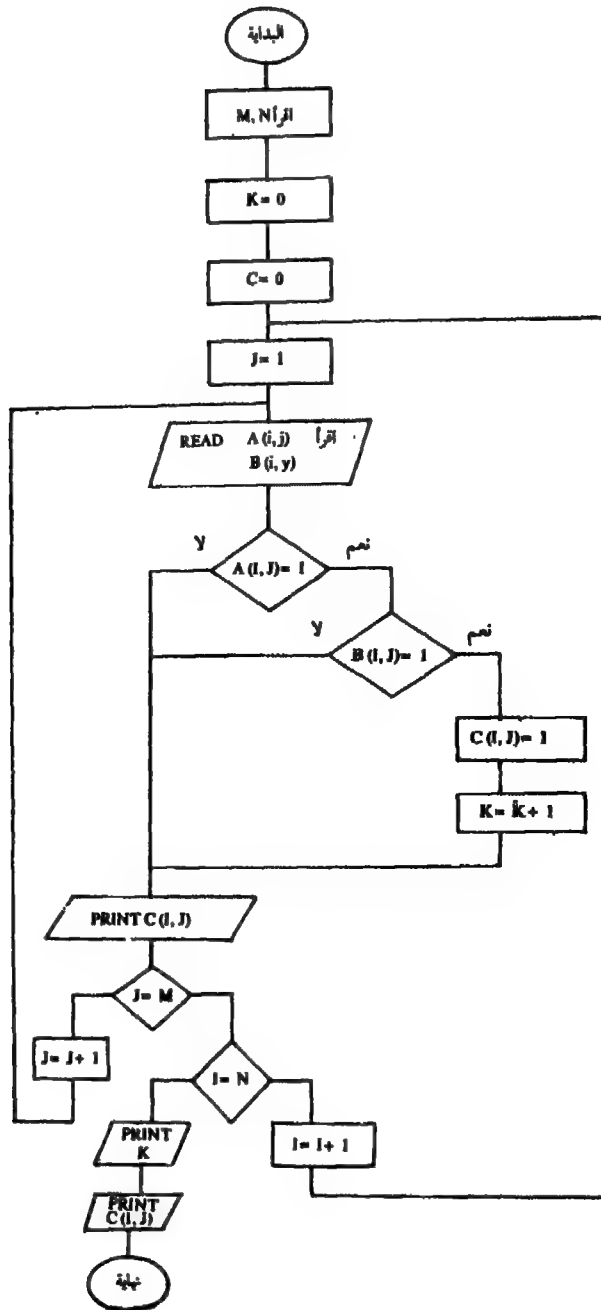
مسألة 5 :

الجدولان A و B هما بنفس الأبعاد ومضمون خلاياهم يساوي واحد (1) أو صفر (0) . اكتب البرنامج الذي يقوم بضرب الجدولين A و B على الشكل التالي :

$$C_{IJ} = A_{IJ} \cdot B_{IJ}$$

ومن ثم يحسب عدد الخلايا في الجدول C والتي تساوي بمضمونها واحد (1) .

التسلسل المنطقي للبرنامج هو :



البرنامج :

```
10 REM PROGRAM MAIN
20 DIM a(20,30),b(20,30),c(20,30)
30 READ m,n
40 k=0
50 i=1
60 j=1
70 READ a(i,j),b(i,j)
80 IF a(i,j)<1 THEN 100
90 IF b(i,j)=1 THEN 120
100 c(i,j)=0
110 GOTO 130
120 c(i,j)=1
130 k=k+1
140 PRINT c(i,j)
150 IF j=m THEN 180
160 j=j+1
170 GOTO 70
180 IF i=n THEN 210
190 i=i+1
200 GOTO 60
210 PRINT k
220 DATA 3,4
230 DATA 0,1,1,1,0,0,1,0,1,1
235 DATA 1,1,0,1,0,1,1,0,0,0,1,1,1,0
240 END
```

مسألة 6 :

اكتب البرنامج الذي يبحث عن العدد الأصغر (S) والعدد الأكبر (L) ، في

لائحة من الأعداد تحتوي على 200 عدد ، تنتهي بالعدد الوهمي 1000 الذي يدل على نهاية اللائحة .

```

100 DATA 3, 4, 15, 17, -4, -5, -2, 0, 17
105 DATA
107 DATA 1000
110 READ x
120 s=x
130 l=x
140 FOR q=2 TO 200
150 READ x
160 IF x=1000 THEN 230
170 IF s<=x THEN 200
180 s=x
190 GOTO 220
200 IF l>=x THEN 220
210 l=x
220 NEXT 'q
230 PRINT "s=";s,"l=";l
240 END

```

مسألة 7 :

اكتب البرنامج الذي يبحث عن العدد الأصغر (S) في لائحة من الأعداد تحتوي على 200 عدد تنتهي بالعدد الوهمي 1000 ، كما ويعين رقم هذا العدد في الجدول (موقع العدد) .

```

100 DATA 3, 4, 15, 17, -4, -5, -2, 0, 17
105 DATA
107 DATA 1000
110 READ x
120 s=x
130 p=1
140 FOR w=2 TO 200
150 READ x

```

```

160 IF x= 1000 THEN 210
170 IF s<=x THEN 200
180 s=x
190 p=w
200 NEXT w
210 PRINT "s=";s,"p=";p
220 END

```

مسألة 8 :

منتجات أحد المصانع هي مُسجّلة في ذاكرة الكمبيوتر على الشكل الآتي :

رقم المنتوجة وسعرها . اكتب البرنامج الذي يقوم بإيجاد سعر المنتوجة بواسطة رقمها . لائحة الأرقام والأسعار تنتهي بالعدد الوهمي 1000 وتحتوي على أرقام وأسعار 200 منتوجة .

الحل :

يقرأ البرنامج كل عددين متتاليين من الأعداد الداخلة إليه بواسطة DATA على حدة ، فيقارن العدد الأول بالرقم الذي يعطيه المسائل (الزبون) ، ويُلقنه للكمبيوتر بواسطة التعليم (INPUT) . وعندما يجد الكمبيوتر العدد الذي يساوي هذا الرقم يكون العدد المقروء الثاني مساوياً لسعر المنتوجة فيطبعه .

```

100 DATA 174,21.51,149,17.20,153,16.30,159,1.25
110 DATA 175,8.60,178,36.75,1000,0
120 RESTORE
130 PRINT"ENTER CATALOG NO.TOBE FOUND"
140 INPUT v
150 IF v=0 THEN 270
160 FOR k=1 TO 200
170 READ x,y
180 IF x=1000 THEN 250
190 IF x=v THEN 220
210 GOTO 240

```

```

220 PRINT x,y
230 GOTO 120
240 NEXT k
250 PRINT " DINTFIND ";v
260 GOTO 120
270 END

```

مسألة 9 :

معنا جدولاً بأسماء وأعمار عدد من الطلاب (عشرة طلاب) . إكتب البرنامج الذي يبحث عن سن كل طالب بواسطة معرفة اسمه .

```

100 DATA TOM, 15, AHMED, 15, JEAN, 13, ZAHRA, 19
110 DATA RANDA, 18, RAMA, 20, RACHA, 18
120 DATA HALA, 16, MAZEN, 13, ALI, 19
130 DIM n$(10), a(10)
140 FOR k=1 TO 10
150 READ n$(k), a(k)
160 NEXT k
170 PRINT "THIS PROGRAM GIVES AGES"
180 PRINT " WHAT IS YOUR NAME"
190 INPUT v$
200 IF v$="stop" THEN 280
210 FOR i=1 TO 10
220 IF v$=n$(i) THEN 260
230 NEXT i
240 PRINT "NAME NOT HERE ";v$
250 GOTO 180
260 PRINT v$; " YOU ARE ";a(i); " YEARSOLD "
270 GOTO 180
280 END

```

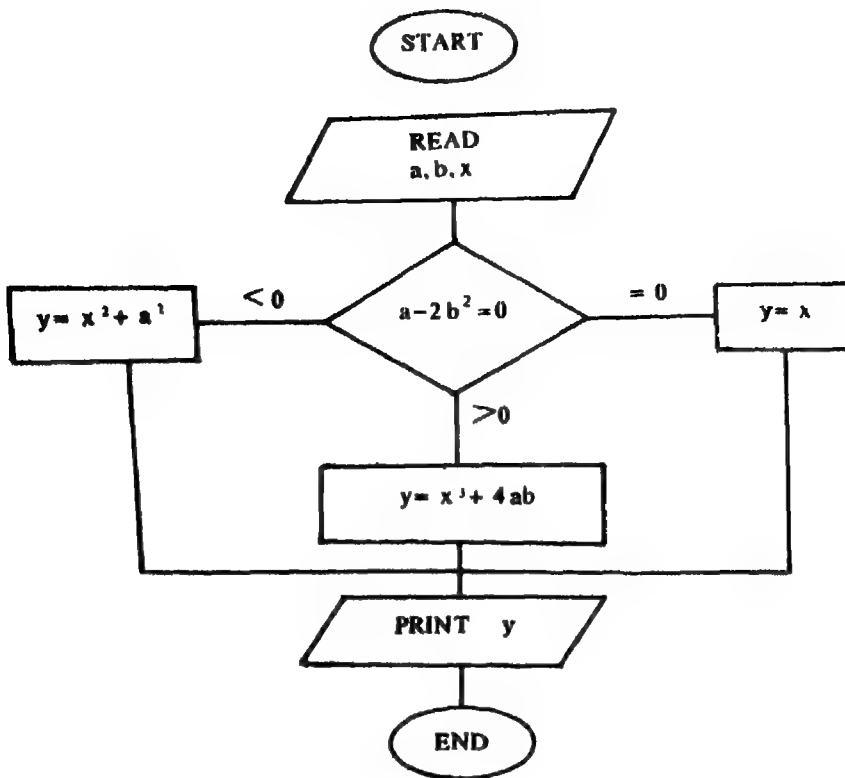
مسألة 10 :

احسب قيمة المتحولة Y التالية ، حسب المعادلة الرياضية التالية :

$y = x^2 + a^2$	if $a - 2b < 0$
$y = x^3 + 4ab$	if $a - 2b > 0$
$y = x$	if $a - 2b = 0$

الحلّ

التسلسل المنطقي للبرنامج :



البرنامج

```
10 REM TRANSFER OF CONTROL EXAMPLE
20 INPUT "a, b, x "; a, b, x
25 d=a-2*b
30 IF d<0 THEN 70
40 IF d>0 THEN 90
50 y=x
60 GOTO 100
70 y=x^2+a^2
80 GOTO 100
90 y=x^3+4*a*b
100 PRINT y
120 END
```

مسألة 11 :

اكتب البرنامج الذي يقوم بطباعة جدول القيم التالية :

$$X_1, X_2 = X^2, X_3 = X^3, X_4 = 1/X_1, X_5 = \sqrt{X}$$

وذلك لأربعين قيمة لـ X .

الحلّ

سنستعمل في هذا البرنامج الحلقة FOR... TO... التي بواسطتها سنقرأ قيمة كل متحولة X ، ومن ثمّ نحسب المتحولات المطلوبة ونقوم بطباعتها .

البرنامج

```
90 MODE 2
100 REM TABULAR VALUE OF X
105 PRINT "x1", "x2", "x3", "x4", "x5=SQR.ROOT"
110 FOR x1=1 TO 40
115 x2=x1*x1
```

```

120 x3=x1^3
130 x4=1/x1
140 x5=x1^0.5
150 PRINT x1,x2,x3,x4,x5
160 NEXT x1
170 END

```

RUN

X ₁	X ₂	X ₃	X ₄	X ₅ = SQR. ROOT
1	1	1	1	1
2	4	8	5	1.41421
.
.
.
40	1600	64000	.025	6.32456

مسألة 12 :

اكتب البرنامج الذي يقوم بإخراج وطباعة جدولاً مؤلفاً من خمسة أعمدة ،
كل عمود يحتوي على درجة الحرارة بالأنظمة التالية :
RANKINE (R), KELVIN (K), CELSIUS (C), FAHRENHET (F)

الحل :

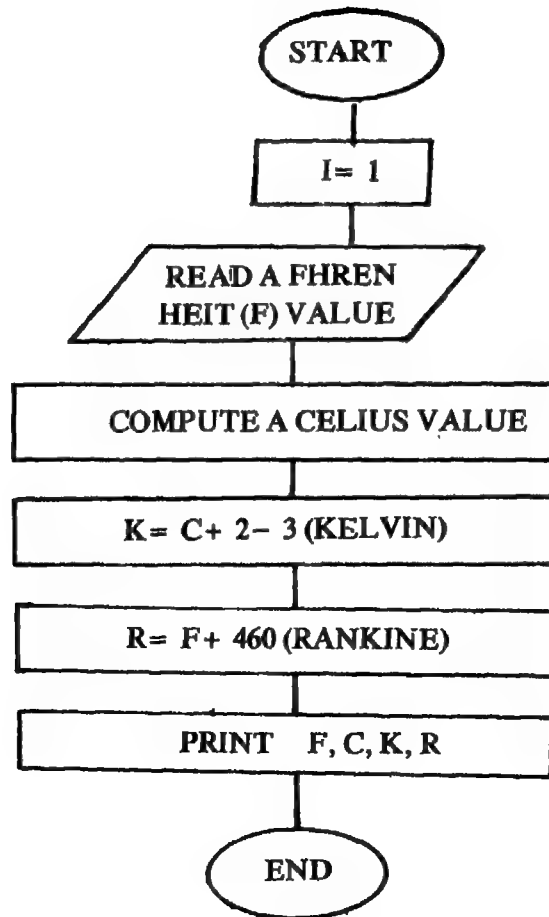
العلاقة بين درجات الحرارة بالأنظمة المطلوبة هي :

- درجة الحرارة بالنظام المثوي تساوي : $C = 519 (F - 32)$

- درجة الحرارة بنظام كالفين تساوي : $K = C + 273$

- درجة الحرارة بنظام رانكين تساوي : $R = F + 460$

التسلسل المنطقي للبرنامج هو :



البرنامج :

لنفترض بأنه معنا عدد من درجات الحرارة يعادل مئة ، أي أن الجدول سيحتوي على مئة سطر بخمسة أعمدة .

```

100 REM TEMPERATORE CONVERSION PROGRAM
110 PRINT " Fah"; TAB(11); "Cel"; TAB(21); "Kel";
    TAB(31); "Ran"
120 PRINT
160 FOR f=15 TO 107
170 c=5/9*(f-32)
  
```

```

180 k=c+273
190 r=f+460
200 PRINT f;TAB(10);ROUND(c,3);
      TAB(20);ROUND(k,3);TAB(30);r
210 NEXT f
230 ENT

```

RUN

FAN	CEL	KEL	RAN
15
18			
19			
.			
.			
.			
107			

مسألة 13 :

اكتب البرنامج الذي يقوم بعملية نسخ المعلومات الموجودة في الجدول A إلى الجدول B .

للجدول A و B نفس العدد من الأعمدة والأسطر .

الحل :

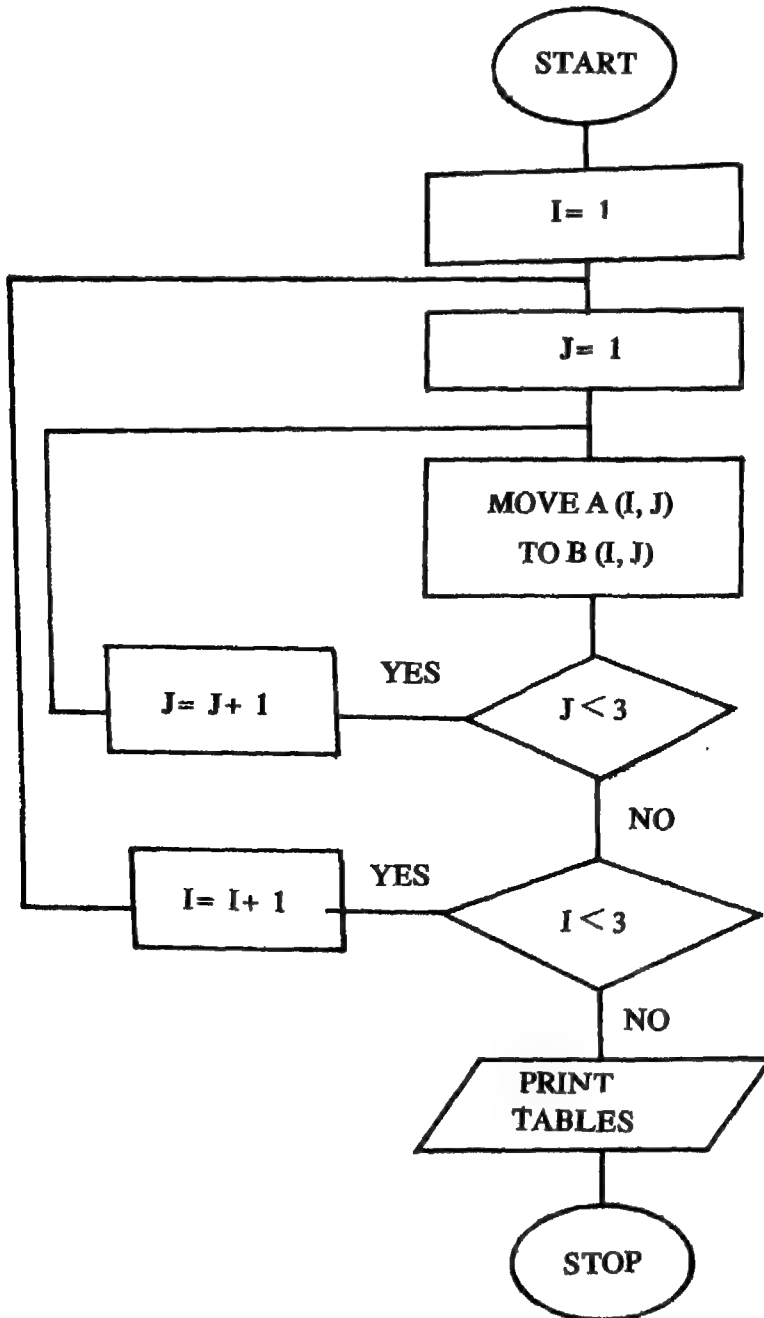
14	16	17
8	9	11
14	200	300

A

	COL 1	COL 2	COL 3
ROW 1			
ROW 2			
ROW 3			

B

التسلسل المنطقي للبرنامج أو الخوارزم :



I عبارة عن مؤشر للأسطر و J عبارة عن مؤشر للأعمدة . بواسطة
الحلقات: FORJ= 1 TO 3, FORI= 1 TO 3 يتم قراءة ونقل جميع خلايا الجدول
A إلى الجدول B .

في البداية $I = 1$ ويبدأ تنفيذ الحلقة الثانية $J = 1$ ويتم نسخ مضمون الخلية
الأولى $A(1, 1) = 14$ إلى الخلية $B(1, 1)$ ، بعد ذلك نزيد قيمة J فتصبح $J = 2$ ،
فيتم نسخ مضمون الخلية الثانية $A(1, 2) = 16$ إلى الخلية $B(1, 2)$. الخ .
بعد ذلك تزداد قيمة I (1) واحد ، فتصبح $I = 2$. ونبدأ من جديد بتنفيذ الحلقة
الثانية ، $J = 1$ ، فيتم نقل مضمون الخلية الأولى من السطر الثاني $A(2, 1)$ إلى
الخلية الأولى من السطر الثاني للجدول B . الخ .

البرنامج :

```

100 REM MOVE TABLE A TO TABLE B
110 DIM a(3,3),b(3,3)
120 FOR x=1 TO 3
130 FOR y=1 TO 3
140 READ a(x,y)
150 b(x,y)=a(x,y)
160 NEXT y
170 NEXT x
180 FOR i=1 TO 3
190 FOR j=1 TO 3
200 PRINT TAB(j*5);b(i,j);
210 NEXT j
220 NEXT i
230 DATA 14,16,17,8,9,11,14,200,300
240 END

```

```

run
14      16      17
8        9       11
14      200     300

```

مسألة 14 :

اكتب البرنامج الذي يحسب الجذر المثلث (CUBEROOT) لعدد ما ، وذلك باستعمال طريقة (NEWTON-RAPHSON) مع العلم بأن الجذر الثلاثي التقريبي Y_0 هو معلوم ، وأن الخطأ المسموح به يجب ألا يتعدى العدد $E = 0.0001$.

الحل :

لنفترض أن الجذر التقريبي هو Y_0 . عند ذلك سيعادل الجذر الثلاثي التالي وبواسطة طريقة NEWTON-RAPHSON ما يلي :

$$Y_1 = \frac{1}{3} \left(\frac{x}{y_0^2} + 2y_0 \right)$$

$$Y_{i+2} = \frac{1}{3} \left(\frac{x}{y_i^2} + 2y_i \right)$$

$$Y_{i+1} - Y_i < E$$

ولنفترض الآن أن العدد المطلوب حسابه جذره هو 106

فالبرنامج الذي يحسب الجذر الثلاثي هو :

```

10 READ x,y,e
20 y1=1/3*(x/y^2+2*y)
30 IF ABS(y-y1) < e THEN 60
40 y=y1
50 GOTO 20
60 PRINT"CUBE ROOT OF ";x;" = ";y1
70 DATA 106,14,.0001
80 END

```

RUN
CUB ROOT OF 106 EQ 4.73262.

مسألة 15 :

اكتب البرنامج الذي يقوم بطباعة جدول للدوال TAN ، COS ، SIN ،
(TRIGONOMETRIC FUNCTION) ، COTAN . لكل زاوية من الزوايا
المعطاة بوحدة RADIANS .

الحل :

بإمكاننا استعمال البرامج الموجودة في مكتبة البرامج لحساب SIN (X) ،
COS (X) ، TAN (X) . أما بالنسبة للدالة COTAN فتحسب على الشكل
التالي : $COTAN(X) = COS(X) / SIN(X)$.

وفي البداية يجب تحويل الزوايا من وحدة الدرجات (DEGREE) إلى الوحدة
RADIAN . وهذا يتم على الشكل التالي :

$$X \text{ radians} = (X \text{ degrés}) \times 180$$

$$LET \ R = D * 3.1415 / 180$$

ولنفترض أن عدد الزوايا هو 100 (نبدأ من الزاوية صفر وحتى الزاوية 100
درجة) فعدد أسطر الجدول سيكون مئة :

البرنامج :

```
10 REM ..TRIGONOMETRIC TABLE PROGRAM..
20 DEF FN c(x)=COS(x)/SIN(x)
30 d=0
40 PRINT"DEGREE SIN COS TAN COTAN"
50 PRINT
60 PRINT " 0 0 1 0 inf"
70 d=d+1
80 REM ..CONVERT DEGREE TO RADIANS..
90 r=d/57.2958
100 PRINT d;TAB(8);ROUND(SIN(r),3);TAB(16);ROUND
(COS(r),3);TAB(24);ROUND(TAN(r),3);TAB(32);ROUND
(FNc(r),3)
```

```

110 IF d<100 THEN 70
120 PRINT
130 PRINT"DEGREE SIN COS TAN COTAN"
140 END

```

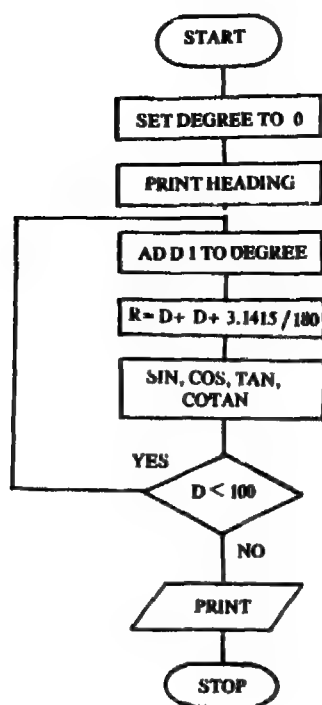
```

RUN
DEGREE SIN COS TAN COTAN

```

0	0	1	0	inf
1	0.017	1	0.017	57.29
2	0.035	0.999	0.035	28.636
4	0.07	0.998	0.07	14.301
5	0.087	0.996	0.087	11.43
6	0.105	0.995	0.105	9.514
7	0.122	0.993	0.123	8.144
8	0.139	0.99	0.141	7.115
9	0.156	0.988	0.158	6.314
10	0.174	0.985	0.176	5.671
11	0.191	0.982	0.194	5.145
12	0.208	0.978	0.213	4.705

خوارزم البرنامج :



مسألة 16 :

اكتب البرنامج الذي يقوم بطباعة نتيجة جمع مصفوفتين هما A, B .

$$\text{MATRIX A} + \text{MATRIX B} = \text{MATRIX C}$$

البرنامج :

سنستعمل الأوامر المستعملة في لغة بازيك والتي تسمح لنا بالعمل مع

المصفوفات من نوع MATRIX .

```

10 DIM c(3,3),a(3,3),b(3,3)
20 FOR i=1 TO 3:FOR j=1 TO 3
30 READ a(i,j):NEXT j:NEXT i
40 FOR i=1 TO 3:FOR j=1 TO 3
50 READ b(i,j):NEXT j:NEXT i
60 CLS
70 PRINT TAB(7);"matrix A";TAB(27);"matrix B"
80 FOR i=1 TO 3:FOR j=1 TO 3
90 LOCATE j*5,i+2:PRINT a(i,j)
100 LOCATE j*5+20,i+2:PRINT b(i,j)
110 NEXT j:NEXT i
120 REM ....add matrices a to b.....
130 FOR i=1 TO 3:FOR j=1 TO 3
140 c(i,j)=a(i,j)+b(i,j):NEXT j:NEXT i
150 PRINT:PRINT: PRINT"Sum Of MATRICES A And B Is
    MATRIX C"
160 FOR i=1 TO 3:FOR j=1 TO 3
170 LOCATE j*5+10,i+10:PRINT c(i,j):NEXT j:NEXT i
200 DATA 2,1,8,9,0,4,114,16,8,4,6,9,-2,3,-1,5,23,

```

run

matrix A

matrix B

2	1	8	4	6	9
9	0	4	-2	3	-1
114	16	8	5	23	10

Sum Of MATRICES A And B Is MATRIX C

6	7	17
7	3	3
119	39	18

مسألة 17 : اكتب البرنامج الذي يقوم بحساب نتيجة ضرب الجداول المربعة A ، B ،
للبرنامج
[C] = [A] × [B]

```

10  DIM      C[ 3,3 ] , A[ 3,3 ] , B[ 3,3 ]
20  MAT      READ A, B
30  PRINT    "MATRIX A"
40  MAT      PRINT A
50  PRINT
60  PRINT    "MATRIX B"
70  MAT      PRINT B
80  PRINT
90  PEM      DETERMINE THE PRODUCT (A)*(B)
100 MAT      C= A* B
110 PRINT    "MATRIX C"
120 MAT      PRINT C
130 DATA    1, 2, 3, ...
140 END

```

RUN

مسألة 18 :

اكتب البرنامج الذي يحسب قيمة X, Y, Z ، ويُحلّ المعادلات الخطية
التالية :

$$X + 2Y + 3Z = 26$$

$$3X + 5Y + 2Z = 39$$

$$2X + 4Y + Z = 27$$

الحل :

هذه المعادلة يمكن كتابتها على الشكل التالي :

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 5 & 2 \\ 2 & 4 & 1 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 26 \\ 39 \\ 27 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix} M \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 26 \\ 39 \\ 27 \end{bmatrix}$$

لحل هذه المسألة ؛ يجب أن نحسب الجدول العكسي (INVERS MATRIX) M^{-1} للجدول M ، (يتم ذلك بواسطة البرنامج 1) . وبعد ذلك نقوم

بعملية ضرب للجدول العكسي M^{-1} بالجدول $\begin{bmatrix} 26 \\ 39 \\ 27 \end{bmatrix}$

لنحصل على قيمة المتحولات X, Y, Z . (برنامج 2) .

برنامج 1 .

```

10  DIM      M[ 3,3] ,D[ 3,3]
20  MAT      READ M
30  MAT      D= INV (M)
40  MAT      PRINT D
50  DATA    1,2,3,3,5,2,2,4,1
60  END

```

RUN

```

- 6.      2      - 2.2
.2      - 1      1.4
.4      0      - .2

```

برنامج 2 .

```

10  DIM      D (3,3), S (3,1),A (3,1)
20  MAT      READ  D, S
30  MAT      A= D* S
40  MAT      PRINT  A
50  DATA    .6, 2, -2.2, .2, - 1 - 1.4, .4, 0, - .2,
60  DATA    26,39,27
70  ED

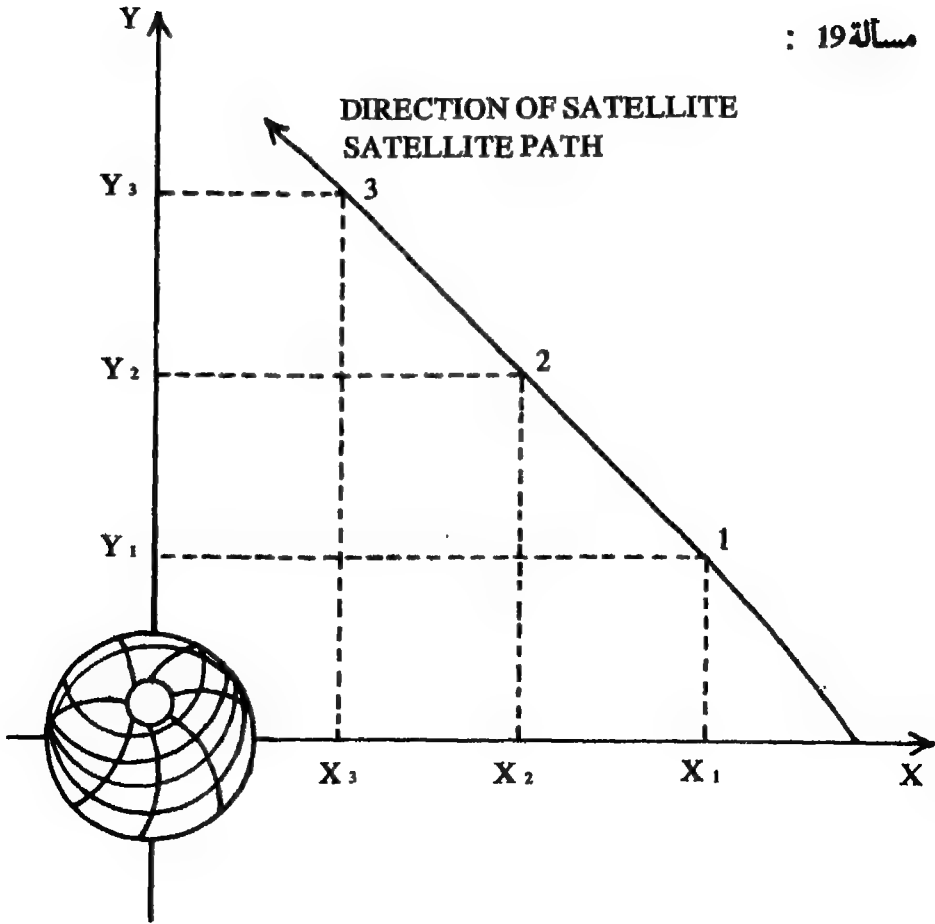
```

RUN

```

3 .
4 .
5 .

```



في هذه الصورة نرى خط مسار قمر اصطناعي بالنسبة للكرة الأرضية . النقاط 1, 2, 3 تحدد موقع القمر بالنسبة للأرض في ثلاثة أزمنة متعادلة . القمر يحتاج لنفس الوقت كي يتحرك من النقطة 2 إلى الموقع 3 ، ومن الموقع 1 إلى الموقع 2 . ويمكن تحديد ومعرفة مكان القمر في الموقع 3 (COORDINATES) إذا عرفنا X_1 ، X_2 ، Y_1 ، Y_2 وفي الموقع 1 و 2 . حسب المعادلة التالية .

$$X_3 = 2X_2 + X_1 \left(\frac{C}{(X_1^2 + Y_1^2)^{3/2}} - 1 \right)$$

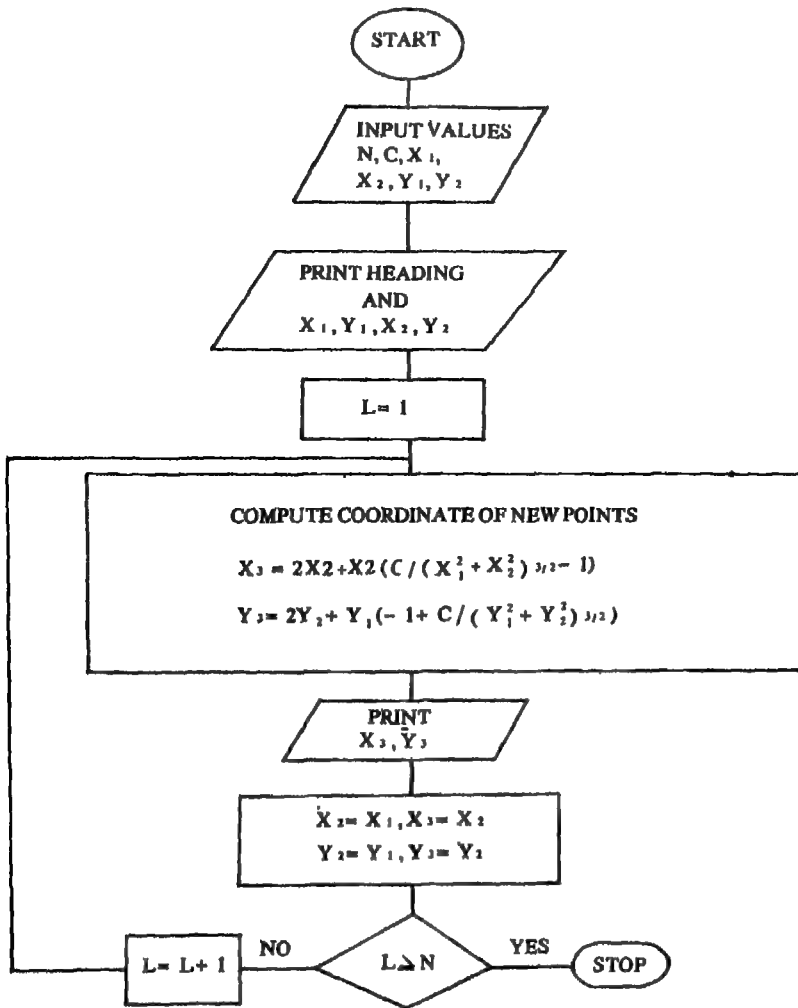
$$Y_3 = 2Y_2 + Y_1 \left(\frac{C}{(X_1^2 + Y_1^2)^{3/2}} - 1 \right)$$

C - عبارة عن ثابتة تتعلق بالجاذبية الأرضية (GRAVITATIONAL ATTRACTION) وبالوقت .

بعد أن يتم حسابة ومعرفة الموقع 3 . نستطيع أن نعرف أي موقع على مسار القمر بنفس الطريقة .

البرنامج يجب أن يقرأ قيمة C و n (n - تساوي عدد المواقع المطلوب حسابة (COORDINATES) و X_1, X_2, Y_1, Y_2 للمواقع الأولى

الخوارزم :



```

10 REM ...SATELLITE ORBIT PROBLEM ...
20 PRINT "TYPE VALUE FOR N,C,X1,Y1,X2,Y2"
30 INPUT n,c,x1,y1,x2,y2
40 PRINT
50 PRINT"COORDINATES OF SATELLITE ORBITE"
60 PRINT
70 PRINT"POINT NOMBRES XCOORDINATE YCOORDINATE"
80 PRINT " 1",x1,y1
90 PRINT" 2",x2,y2
100 FOR k=1 TO n
110 x3=2*x2+x1*(c/((x1^2+y1^2)^1.5-1))
120 y3=2*y2+y1*(c/((x1^2+y1^2)^1.5-1))
130 PRINT k+2,x3,y3
140 x1=x2
150 x2=x3
160 y1=y2
170 y2=y3
180 NEXT k
190 END

```

run

```

run
TYPE VALUE FOR N,C,X1,Y1,X2,Y2
? 4,5,10000,20000,35000,40000

```

```

COORDINATES OF SATELLITE ORBITE

POINT NOMBRES XCOORDINATE YCOORDINATE
1              10000      20000
2              35000      40000
3              70000      80000
4             140000     160000
5             280000     320000
6             560000     640000
Ready

```

شروحات حول البرنامج :

التعليمة 30 ، يقوم بإدخال المعطيات الثابتة للبرنامج وهي X_1 ، C ، N ،
 Y_1 ، Y_2 . الخمسة تعليمات التالية تقوم بطباعة عنوان جدول (تحرك القمر) .
 التعليمات 80 ، 90 ، تقوم بطباعة أول سطرين من الجدول وهي المواقع واحد
 واثنين. التعليمات التالية 180 ÷ 100 تقوم بحسابه وتحديد المواقع التالية وعددها N .
 الاوامر 160 ÷ 140 تقوم بتغيير النقاط المستعملة لتحديد موقع النقطة التالية .

مسألة 20 :

اكتب البرنامج الذي يحسب قيمة البولينوم (POLYNOME) لجميع الأعداد
 الصحيحة X الموجودة بداخل القسم (16 + 12 -) .
 $- 12 < X < + 16$

كما ويحسب الجذر X في الحالة $y = 0$ للمعادلة $y(x)$.

$$y(X) = y = X^6 - 3X^5 - 93X^4 + 87X^3 + 1596X^2 - 1380X - 2800$$

كما ويطبّع القيمة العليا والسفلى P وقيمة X في هذه النقاط .

شروحات حول البرنامج :

الأمر 50 يحسب قيمة Y لجميع المتحولات X من العدد 11 - وحتى 16

الأمر 60 يسأل إذا كانت $Y = 0$.

الأمر 80 يسأل هل ان $X = 11$ ، أي نقطة البداية .

الأمر 120 يسأل هل $Y_1 < Y$ ، وفي الحالة التي تكون فيها $Y_1 < Y$

نذهب إلى التعليمة 160 وإلا $X_1 = X$ ، $Y_1 = Y$.

التعليمة 160 يسأل هل $Y_1 > Y$ فإذا لم تكن هذه المعادلة حقيقية سيكون معنا

، $Y_2 = Y$ ، $X_2 = X$ وإلا نقوم بطباعة الجمل التالية .

POLYNOMIAL IS SMALLEST WHEN $X = "$ ، X_1

POLYNOMIAL IS LARGEST WHEN $X = "$ ، X_2

البرنامج :

```
10 REM ..POLYNOMIAL EVALUTION PROGRAM..
20 PRINT "Zeros Values Of X When The Polynomial Is"
30 PRINT
40 FOR x=-11 TO 16
50 y=((((x-3)*x-93)*x+87)*x+1569)*x-1380)*x-2800
60 IF y<>0 THEN 80
70 PRINT "p=0 When X= "; x
80 IF x<>-11 THEN 120
90 x1=x
100 y1=y
110 GOTO 170
120 IF y1<y THEN 160
130 x1=x
140 y1=y
150 GOTO 190
160 IF y1>y THEN 190
170 x2=x
180 y2=y
190 NEXT x
200 PRINT "POLYNOMIAL IS SMALIEST WHEN X="; x1
210 PRINT "POLYNOMIAL IS LARGEST WHEN X="; x2
220 END
```

run

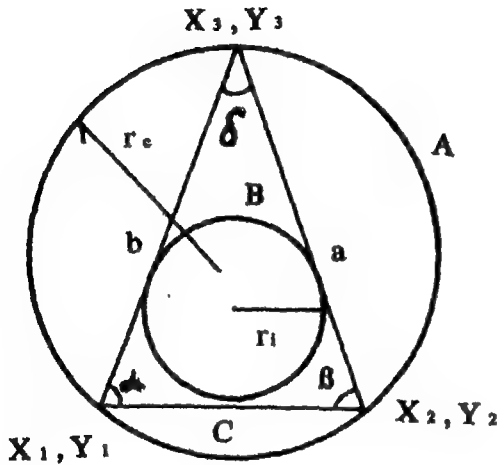
Zeros Values Of X When The Polynomial Is

POLYNOMIAL IS SMALIEST WHEN X= 8

POLYNOMIAL IS LARGEST WHEN X= 16

Ready

مسألة 21 :



معنا ثلاثة نقاط بأبعاد (Y_1, X_1) ، (Y_2, X_2) ، (Y_3, X_3) . تؤلف
 مثلثاً موجوداً بداخل دائرة (الدائرة A) ، أما الدائرة B فموجودة بداخله .
 r_0 شعاع الدائرة الخارجية التي تمر بالنقاط الثلاثة .
 r_1 شعاع الدائرة الداخلية الموجودة بداخل المثلث (انظر الصورة) .
 c, b, a أطراف المثلث .

المطلوب :

- حساب قيمة الأطراف a, b, c

- مساحة المثلث

- الشعاع r_1

- الشعاع r_0 .

- الزوايا B, α

```

10 REM .....GEOMETRY PROGRAM.....
20 PRINT"TYPE COORDINATES OF TRINGLE IN THE"
30 PRINT"FOLLOWING ORDER : X1,Y1,X2,Y2,X3,Y3"
40 INPUT x1,y1,x2,y2,x3,y3
50 REM ....COMPUTE LEN THS OF SIDES....
60 a=SQR((x3-x2)^2+(y3-y2)^2)
70 b=SQR((x3-x1)^2+(y3-y1)^2)
80 c=SQR((x2-x1)^2+(y2-y1)^2)
90 REM ....COMPUTE AREA....
100 s=(a+b+c)/2
110 a1=SQR(s*(s-a)*(s-b)*(s-c))
    
```



```

      r1=a1/s
130 REM .COMPUTE RADIUS OF CIRCUS CRIPED CIRLE.
140 r2=a*b*c/(a1*4)
150 REM .....COMPUTE ANGLES.....
155 IF b^2+c^2-a^2=0 THEN x=PI/2 :GOTO 165
160 x=ATN((4+a1)/(b^2+c^2-a^2))
165 IF a^2+c^2-b^2=0 THEN y=PI/2 :GOTO 175
170 y=ATN((4*a1)/(a^2+c^2-b^2))
175 IF a^2+b^2-c^2=0 THEN z=PI/2:GOTO 190
180 z=ATN((4+a1)/(a^2+b^2-c^2))
190 REM .PRINT COORDINATES OF TRIANGLE.
200 PRINT"COORDINATES OF TRIANGLE."
210 PRINT"x1=";x1,"y1=";y1
220 PRINT"x2=";x2,"y2=";y2
230 PRINT"x3=";x3,"y3=";y3
240 PRINT
250 REM ...PRINT CALCULATED VALUES...
260 PRINT"a=";a;"    b=";b;"    c=";c
270 PRINT"AREA OF TRIAGLE IS";a1
280 PRINT"RADIUS Ri IS";r1
290 PRINT"RADIUS Rc IS";r2
300 PRINT"ALPHA=";x
310 PRINT"BETA=";y
320 PRINT"GAMA=";z
330 END

```

run

TYPE COORDINATES OF TRINGLE IN THE
FOLLOWING ORDER : X1,Y1,X2,Y2,X3,Y3? 1,1,1,2,2,1

COORDINATES OF TRIANGLE.

```

x1= 1          y1= 1
x2= 1          y2= 2
x3= 2          y3= 1

```

```

a= 1.41421356    b= 1    c= 1
AREA OF TRIAGLE IS 0.5
RADIUS Ri IS 0.292893219
RADIUS Rc IS 0.707106781
ALPHA= 1.57079633
BETA= 0.785398163
GAMA= 0.785398163

```

مسألة 22 :

اكتب البرنامج الذي يحسب العلاقة الرياضية التالية :

$$\binom{n}{K} = \frac{n!}{K!(n-K)!}$$

الحل

من المعلوم إن :

$$\binom{n}{1} = n$$

لنحسب $\binom{n}{j}$ لعدد K من القيم J ، أو :

FOR $J = 2, 3, 4, \dots, K$

بنفس الطريقة ، ولو استعملنا المعادلة الموجودة في المسألة سنرى إن :

$$\binom{n}{J} = \binom{n}{J-1} \left(\frac{n+1-J}{J} \right)$$

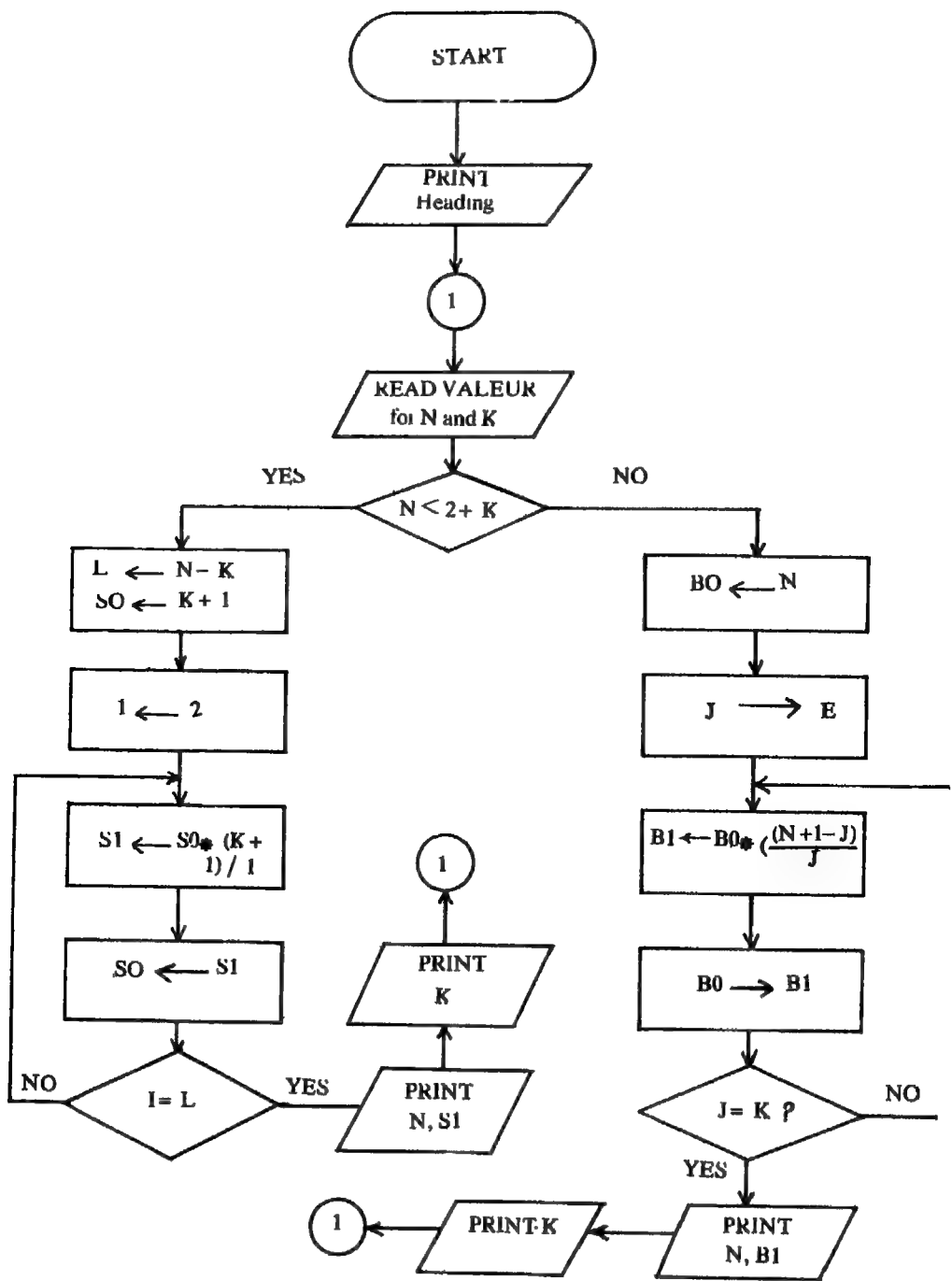
لو افترضنا إن $n < 2K$, $I = 1$ فعملية البداية :

$$\binom{K+1}{K} = \binom{K+1}{K} = K+1$$

بنفس الطريقة فلنحسب

FOR $I = 2, 3, 4, \dots, n-K$

$$\binom{K+I}{K} = \binom{K+I-1}{K} \left(\frac{K+I}{I} \right)$$



BASIC Programming

```
30 REM BINOMIAL COEFFICIENT COMPUTATION
40 PRINT "N Binomiale";
50 PRINT"  k Koefficient"
60 PRINT
70 READ n,k
80 IF n<2*k THEN 180
90 b0=n
100 FOR j=2 TO k
110 b1=b0*((n+1-j)/j)
120 b0=b1
130 NEXT j
140 PRINT n,b1
150 PRINT k
160 PRINT
170 GOTO 70
180 l=n-k
190 s0=k+1
200 FOR i=2 TO l
210 s1=s0*(k+i)/i
220 s0=s1
230 NEXT i
240 PRINT n,s1
250 PRINT k
260 PRINT
270 GOTO 70
280 DATA 9,4,11,6,100,70
290 END
```

```
run
N Binomiale  k Koefficient

9              126
4

11             462
6

100            2.93723E+25
70
```

البرنامج الموضوع يحسب (9 / 4) ، (11 / 6) ، (100 / 70) ويبرهن على أن (100 / 70) هي القيمة العليا .

مسألة 23 :

هذه المسألة تحل معادلة ترانزستور يعمل كمقوي (Amplifier) تحت إشارات ضعيفة (SMALL SIGNAL) وشبكة الدائرة (NET WORK CIRCUITS) موضوعة كتابياً من المهندس المسؤول ، وهي عبارة عن معلومات مطلوب تلقيها للبرنامج .

* GERMANIUM TRANSISTOR

* MINIMUM	design temp	= - 30° C
* MAXIMUM	design temp	= 60° C
* MINIMUM	beta	= 50
* MAXIMUM	beta	= 150
* Smply	Voltage	= 10 V
* EMITTER	sesistor	= 200
* RESISTANCE	TOLERACE	= 5 %
* VCE	= 5 V	
* ICB0	= 0,002 m A (min)	
* ICB0	= 0,78 m A (max)	
* IE	= 1,18 mA	

PROGRAM Listing.

```

70      PRINT      "FOR SILICON TRANSISTOR, ENTER 1,
                  FOR GERMANIUM, ENTER 0"

80      INPUT      S

90      PRINT "ENTER MIN.  MAX. DESIGN TEMPERATURES,
                  MIN.  MAX. BETA

100     PRINT      T1, T2, H1, H2

```

```

110 PRINT "ENTER SUPPLY VOLTAGE, EMITTER
RESISTOR, RES. TOLERANCE IN%"
120 INPUT V0, R4, P
130 PRINT "ENTER V (CE), I (CB0) IN MA, MIN. MAX.
I (E) IN MA "
140 INPUT V1, I0, I1, I2
150 LET I1 = I1 * (1 + 0.3 * P)
160 LET I2 = I2 * (1 - 0.3 * P)
170 LET H1 = H1 * .865 * Exp(.00575 * T1)
180 LET T3 = T2 - 25
190 LET H2 = H2 * (.865 * Exp(.00575 * T2) - (S - 1) *
(.00895 - .00565 * T3 + .0048 * T3 + 2))
200 IF S = 1 THEN 220
210 LET I0 = I0 * Exp(.075 * T3)
220 LET R3 = .000 * V0 - V1 / (I1 + I2) - R4
230 LET R6 = ((I2 - I1) * R4 + 2.5 * (T2 - T1)) / (I0 + I1 /
(H1 + 1) - I2 / (H2 + 1))
240 IF R6 > 0 THEN 270
250 PRINT "I (E) RANGE TOO NARROW"
260 STOP
270 LET V6 = I1 * .001 * (R6 / (H1 + 1) + R4)
* .2 + .5 * 5 - .00 * (T1 - 25)
280 PRINT
290 PRINT "BIAS VOLTAGE = "; V6; "VOLTS
300 PRINT "BIAS RESISTOR = "; R6; "OHMS"
310 PRINT
320 PRINT "FOR STABILIZED BIAS CIRCUIT:
330 PRINT "R (B - VCC) = "; V0 * R6 / V6

```

```

340 PRINT "R (B- GND)= "; V $\phi$ * R6 / (V $\phi$  - V6)
350 PRINT "R (COLL.)= "; R3
360 PRINT "R (EMITTER)= "; R4
370 END

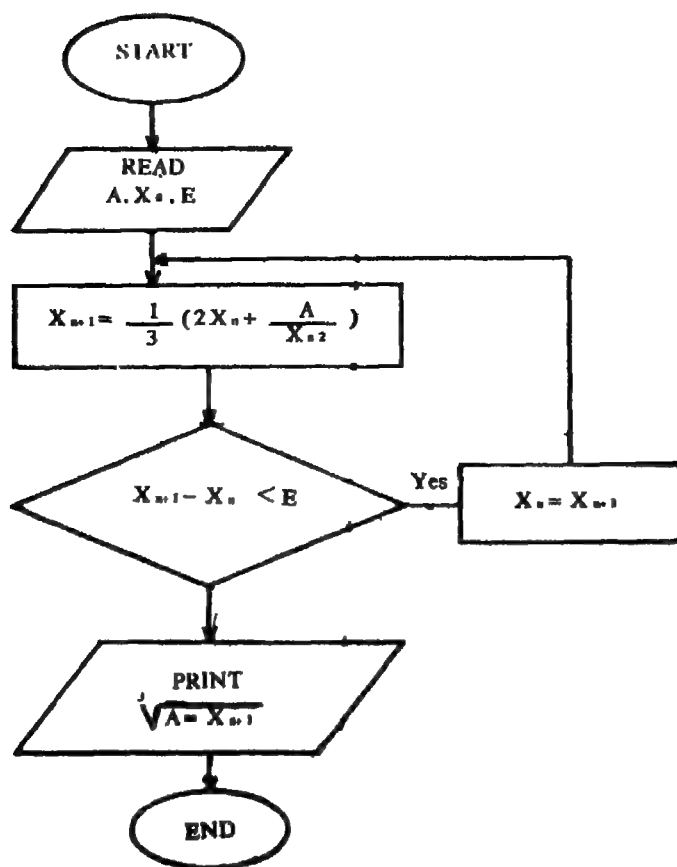
```

مسألة 24 :

اكتب البرنامج الذي يحسب الجذر الثلاثي Cubic root لعددها هو A بواسطة
المعادلة التابعية التالية :

$$X_{n+1} = \frac{1}{3} \left(2X_n + \frac{A}{X_n^2} \right)$$

وبخطأ لا يتجاوز (E = 0.0001) . E . وبقيمة أولية هي X_0 .



```

100  READ      A, X , E
110  DATA    18, 2, .0001, 213, 6, .0001
120  LET       $X_1 = (1 / 3) * (2 * X + A / X * 2)$ 
130  IF       $ABS(X_1 - X) < E$  THEN 160
140  LET      X = X1
150  GOTO     120
160  PRINT    "CUBIC ROOT OF" ; A; "E"; K1
170  GOTO     100
180  END

```

مسألة 25 :

معنا عشرة مجموعات من النقود الذهبية ، تحتوي كل مجموعة على عشرة ليرات ذهبية تزن الواحدة منهم 10 غرامات ، إلا مجموعة واحدة حيث يقل وزن الليرة بـ 0.1 غرام عن غيرهم (أي إن وزن الليرة الواحدة من هذه المجموعة هو 9.9 غرام) كيف يمكننا باستعمال ميزان وبوزنه واحدة فقط ، من أن نكتشف المجموعة التي تحتوي على الليرات الذهبية الخفيفة .

الحل :

10 نقود	10 نقود
0 0 0	0
1 2 3	10

رقم المجموعات

نأخذ من المجموعة الأولى ليرة واحدة ، ومن الثانية ليرتين ، ومن الثالثة 3 ليرات . . ومن العاشرة 10 ليرات . بعد ذلك يتم وزنهم . ولنفترض بأنه لا توجد نقود خفيفة الوزن عند ذلك سيكون وزن النقود هو :

$$(1 + 2 + 3 + 4 + 5 + 5 + 6 + 7 + 8 + 9) \cdot 10 = 450 \text{ g}$$

ولو افترضنا الآن بأن النقود الخفيفة موجودة في المجموعة رقم ثلاثة ، فعند ذلك

سيكون وزن النقود هو :

$$(1 + 2 + 2,97 + 4 + 5 + 6 + 7 + 8 + 9) \cdot 10 = 447 \text{ g}$$

والفرق N هو :

$$N = 450 - 447 = 3$$

فإذا تدلّ N على رقم المجموعة التي تحتوي على النقود الخفيفة .
ولنفترض الآن بأن النقود الخفيفة موجودة في المجموعة السادسة . عند ذلك
سيكون وزن النقود هو :

$$(1 + 2 + 3 + 4 + 5 + 5,4 + 7 + 8 + 9) \cdot 10 = 444$$

$$N = 450 - 444 = 6$$

أو N تساوي رقم المجموعة التي تحتوي على النقود الخفيفة .
والبرنامج هو :

```
10  LET      S = 0
20  LE       N = 1
30  IF       N > 9 THEN '0
40  LET      S = (S + N)* 10
50  LET      N = N + 1
60  GOTO     30
70  INPUT    M
80  LET      R = S - M
90  PRINT    THE lowest ensemble is: ; R
100 END
```

هذا البرنامج يقوم بجمع الأعداد من واحد إلى تسعة وضربهم بعشرة . بعد ذلك يتم انتظار نتيجة الميزان (INPUT M) فيقوم بعملية حسابة الفرق $R = S - M$ ، لنحصل على رقم المجموعة المطلوبة .

مسألة 26 :

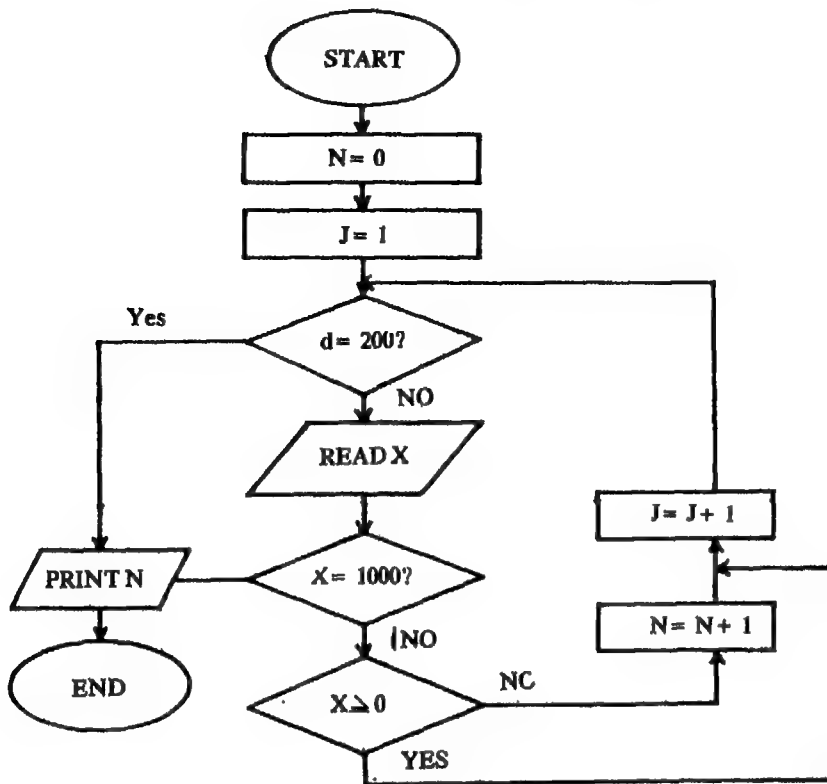
اكتب البرنامج الذي يحسب عدد الأعداد السلبية . في لائحة من الأعداد تحتوي على 200 عدد ، مَصْرُح عنها بواسطة الأمر DATA وتنتهي بالعدد 1000 .

```

100 DATA 21, 17, 0, - 4, - 9, 46, 0, - 2, 0, - 8, 47, 1000
110 LET N = 0
120 FOR J = 1 TO 200
130 READ X
140 IF X = 1000 THEN 180
150 IF X >= 0 THEN 170
160 LET N = N + 1
170 NEXT J
180 PRINT N
190 END

```

والتسلسل المنطقي لهذا البرنامج هو :



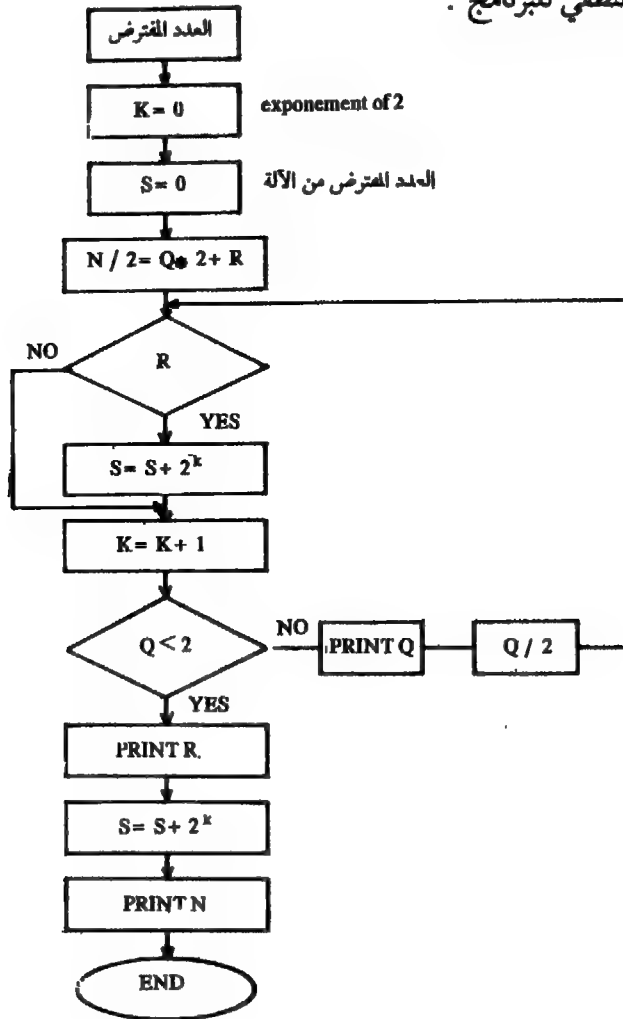
مسألة 27 :

لعبة الأعداد مع الآلة .

يفترض اللاعب عدداً معيناً هو N ($2 < N < 1000$) ، ويجب على الآلة أن تعرف العدد المفترض .

اكتب البرنامج الذي بواسطته تسأل الآلة فتجيبها ، حتى تُدرك الأخيرة العدد المفترض من السائل والذي فُكّر به ، فتقوم بإخراجه وطبعه .

التسلسل المنطقي للبرنامج :



شروحات حول البرنامج :

$$N / 2 = Q.2 + R$$

تقوم الآلة بسؤال اللاعب . « لو قسمنا العدد على 2 ، هل هناك باقي (R) » . فيجيب اللاعب « بنعم » أو « بلا » . ففي حالة الإجابة بنعم تزيد الآلة من قيمة العدد K بواحد . وبعد ذلك تسأل هل نتيجة القسمة Q أكبر من 2 ، فإذا كانت $(Q > 2)$ يجيب اللاعب بحرف P وإلا بحرف R إلخ . بهذه الطريقة تقوم الآلة بإيجاد قيمة العدد بالنظام الثنائي ، وبذلك يكون العدد العشري المفترض هو $S + S \cdot 2^k$

البرنامج :

```
10 PRINT "Ce programme peut connaître le
      nombre que vous pensiez"
15 PRINT "Les regles de la joue"
20 PRINT "SI VOTRE REPONSE AU QUESTION:
      YA-T-IL UN RESTE POSITIVE
      ECRIVEZ : 1"
30 PRINT "SI, VOTRE REPONSE AU QUESTION:
      YA-T-IL UN RESTE EST NEGATIVE
      , ECRIVEZ: 0"
40 PRINT "SI LE QUOTIENT EST PLUS
      GRAND QUE 2, ECRIVEZ: Q"
50 PRINT "SI LE QUOTIENT EST PLUS
      QUE 2, ECRIVEZ : R" PETIT
60      "COMMENCONS A JOUER, PRENEZ
      UN STYLO ET UN LIST"
70 PRINT
80 PRINT "PENSEZ UN NOMBRE DANS
      L'INTERVALE 2 A 1000"
90 LET K = 0
100 LET S = 0
```

```

110 PRINT "DIVISEZ LE NOMBRE SUR 2"
120 PRINT "EST-CE QU'IL Y A UN RESTE"
130 INPUT A
140 IF A = 1 THEN 160
150 IF A = 0 THEN 170
170 LET S = 5 + 2 ↑ K
180 PRINT "LE QUOTIENT REÇU EST
PLUS PETIT QUE 2?"
190 INPUT B
200 IF B = Q THEN 220
210 IF B = R THEN 240
220 PRINT "DIVISEZ LE QUOTIENT
RECU SUR 2 "
230 GOTO 120
240 LET S = 5 + 2 ↑ K
250 PRINT S
260 PRINT "MERCI BEAUCOUPS"
270 END

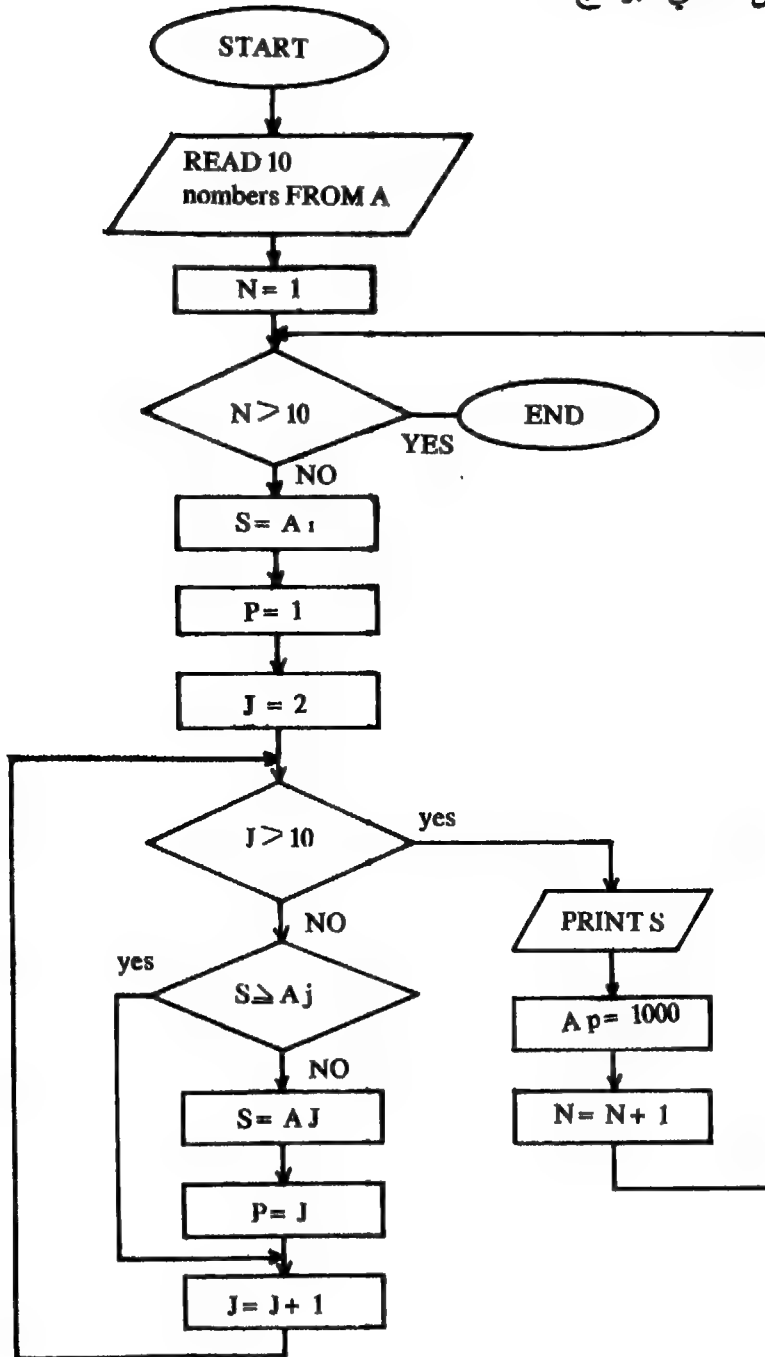
```

هذا البرنامج هو من البرامج التي تعمل في الوقت الحالي "real time"، أي هناك حوار بين الآلة والسائل حتى نحصل على النتيجة المطلوبة .

مسألة 28 :

معنا لائحة من الأعداد تحتوي على عشرة أعداد . اكتب البرنامج الذي يقوم بترتيب هذه اللائحة بشكل تصاعدي للأعداد .

التسلسل المنطقي للبرنامج :



```

100 DIM A (10)
110 DATA 4, 9, 7, - 1, - 4, 18, 14, 13, 4, 7
120 FOR K = 1 TO 10
130 READ A (K)
140 NEXT K
150 FOR N = 1 TO 10
160 LET S = A (I)
170 LET P = 1
180 FOR J = 2 TO 10
190 IF S <= A (J) THEN 220
200 LET S = A (J)
210 LET P = J
220 NEXT J
230 PRINT S
240 LET A (P) = 1000
250 NEXT N
260 END

```

المؤشر N - يدل على الأعداد في اللائحة ويستعمل لقراءتهم . أما المؤشر J - فيستعمل للحصول على العدد الأصغر وبالتالي إعادة ترتيب اللائحة .
مسألة 29 :

معنا ثلاثة أشكال هندسية ؛ دائرة ، مربع ، ومستطيل (rectangle) .
اكتب البرنامج الذي يحسب مساحة هذه الأشكال .
شروحات حول البرنامج :
تصميم البرنامج هو عام ، أي أن البرنامج يحسب مساحة هذه الأشكال حسب
رغبة المبرمج في تلقيم المعلومات المتعلقة بأطرافها (متحولات) .
إدخال المعلومات يتم بواسطة الأمر DATA وعلى شكل لائحة من الأعداد .
العدد الأول يختص بكود (Code) الشكل الهندسي المطلوب حساب مساحته وذلك
كما يلي :

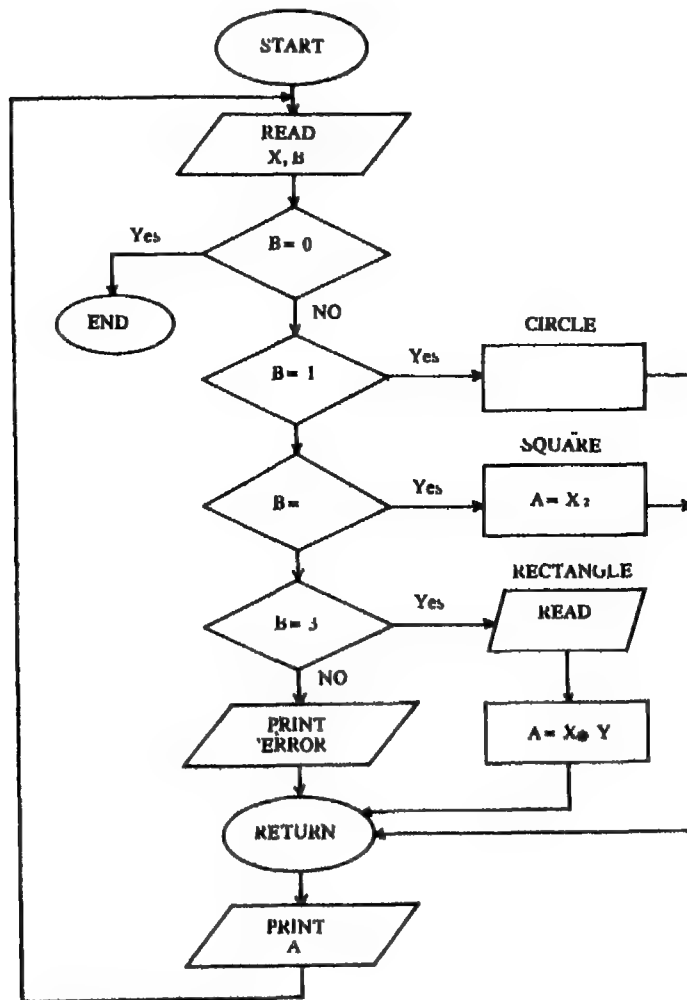
B = 1 - يعني إن الشكل الهندسي هو دائرة CIRCLE .

B = 2 - يعني مربع SQUARE .

B = 3 - ويعني إن الشكل الهندسي هو مستطيل RECTANGLE .

الأعداد التالية تختص بالمتحولات X (أطراف الأشكال) التي تحدد الأشكال شعاع الدائرة أو أطراف المربع . . (تقرأ الآلة هذه الأعداد (المتحولات) فتدعو برنامج الثانوي الخاص الذي يحسب مساحة الشكل الهندسي .

التسلسل المنطقي للبرنامج :



البرنامج :

```
10 DATA 6.1, 1, 2.9, 2, 4.6, 3, 4.5, 1.5, 2
20 DATA 4.9, 3, 8.8, 0.0
30 READ X, B
40 IF B = 0 THEN 220
50 GOSUB 100
60 PRINT A
70 GOTO 30
100 IF B = 1 THEN 150
110 IF B = 2 THEN 170
120 IF B = 3 THEN 190
130 PRINT ERROR IN DATA
140 GOTO 210
150 LET A = 3.1416 * X * 2
160 GOTO 210
170 LET A = X * 2
180 GOTO 210
190 READ Y
100 LET A = X * Y
210 RETURN
220 END
```

مسألة 30 :

اكتب البرنامج الذي يقرأ اسم الشكل الهندسي ، فيحسب مساحته بعد أن يقرأ أطرافه (متحولات الشكل) .

شروحات :

يجب على الآلة أن تفهم أولاً الشكل الهندسي المطلوب حساب مساحته (دائرة ، مربع ، مثلث ، مستطيل) بعد ذلك تقرأ المتحولات الخاصة بالأطراف.

(شكل الشكل الهندسي) فتدعو البرنامج الثانوي الذي يحسب مساحته .

```

10 DATA 1, 5.7, CIRCLE, 2, 7.4, 8.5, RECTANGLE
20 DATA 3, 2.7, 8.5, TRIANGLE, 4, 7.6, SQUARE
30 DATA 2, 4.6, 8.8, RECTANGLE, 4, 5.8, SQUARE
40 DATA 3, 9.4, 7.4, TRIANGLE, 5
50 READ X
60 ON X THEN 70, 110, 150, 190, 230
70 READ R, X $
80 LET 3 = 3.1416 * R
90 PRINT A, R, " ", X $
100 GOTO 50
110 READ L, W, X $
120 LET A = L * W
130 PRINT A, L, W, X $
140 GOTO 50
150 READ B, H, X $
160 LET A = (B * H) / 2
170 PRINT A, B, H, X $
180 GOTO 50
190 READ S, X $
200 LET A = S ^ 2
210 PRINT A, S, " ", X $
220 GOTO 50
230 END

```

مسألة 31 :

لنفترض بأننا نريد أن نسأل الآلة سؤالاً ما ونطلب مساعدتها بواسطة (HELP) . تقوم الآلة بتحليل السؤال وتجييب : أقدر على مساعدتكم (I CAN)

(I DON'T UNDERSTAND HELP YOU) أو لا أقدر على مساعدتكم
 . YOU)

البرنامج :

```

10  DIM      M (100)
20  PRINT    "I AM A COMPUTER PSYCHIATRIST"
30  PRINT    "WHAY DID YOU CONTACT ME?"
40  INPUT    A $
50  CHANGE   A $ TOM
60  FOR      K= 1    TO M (0)- 3
70  IF       M (K)= 72 THEN 90
80  GOTO     140
90  IF       M (K+ 1)= 69 THEN 110
100 GOTO     140
110 IF       M (K+ 2)= 76 THEN 130
120 GOTO     140
130 IF       M (K+ 3)= 80 THEN 170
140 NEXT     K
150 PRINT    "I DON'T UNDERSTAND YOU
160 GOTO     180
170 PRINT    "I CAN HELP YOU
180  END
  
```

مسألة 32 :

البرنامج التربوي الذي يقوم باختبار الطلاب ..
 لنفترض ثلاثة معادلات يجب على الطالب الاجابة عليها وهي :

$$A_1 = X * Y$$

$$B_1 = X / Y$$

$$C_1 = X \uparrow 2 + Y \uparrow 2$$

تسأل الآلة الطالب السؤال الأول : اكتب المعادلة A . فيكتب الطالب معادلته ويُعطِيها للآلة . تفحص الآلة جوابه ، فإذا كان صحيحاً تقول له بأن الجواب هو صحيح وله علامة معينة ، وإلا ترد بأن جوابه خطأ ، وبالتالي تسأله السؤال الآخر ... الخ .

عملية فحص الجواب تتم بحسابة قيمة المعادلة كما هي مُسجَّلة في الذاكرة المركزية للآلة (المعادلة الأكيدة) وقيمة المعادلة التي يجاوب بها الطالب بإعطاء نفس المعطيات للمتحويلات المستعملة في هذه المعادلات بواسطة (DATA) . فإذا كانت النتيجة متطابقة ، فمعناه إن جواب الطالب صحيحاً وإلا يكون جوابه خطأ .

البرنامج :

```

0      GOTO 10
1      ON I GOTO 140, 230, 330
10     READ X, Y
20     DATA .3, .56
30     A1 = X * Y
40     B1 = X / Y
50     C1 = X ↑ 2 + Y ↑ 2
90     I = 1: REM I  Nombre des erreurs
100    HOME: VTAB 5 : HTAB 10
110    PRINT  ECRIVEZ LA FORMULE DE A
115    WRITE EQU. OF A
120    GO SUB 1000
130    END
140    IF A < ' > A1 THEN PRINT: PRINT
      : PRINT 'REPONSE FAULT
      : PRINT 'ERROR EQUATION :
      GOTO 160
150    W = W + 1 : PRINT: PRINT: PRINT

```

```

      REPONSE VRAIE :
      : NO ERROR
160   I= I+ 1
165   GO SUB 1100
150   HOME : VTAB 5
200   PRINT ECRIVEZ LA FORMULE DE B
      : PRINT "WRITE EQU. B
210   GO SUB 1000
220   END
230   IF B <  > B : THEN PRINT: PRINT:
      PRINT "REPONSE FAUT, ERROR
      : GOTO 250
240   W= W+ 1 : PRINT: PRINT: PRINT
      REPONSE VRAIE, NO ERROR
250   I= I+ 1
255   GO SUB 1100
260   HOME : VTAB 5
300   PRINT ECRIVEZ LA FORMULE DE C
310   GO SUB 1000
320   END
330   IF C <  > C : THEN PRINT: PRINT
      : PRINT "REPONSE FAULT":
      GOTO 350
340   W= W+ 1 : PRINT: PRINT: PRINT
      "REPONSE VRAIE"
350   I= I+ 1
355   GO SUB 1100
360   GO TO 2000.

```

```

1000  FOR J= 1 TO 40
1010  XX= PEEK (- 16336):NEXT J
1020  VTAB 10: HTAB 5 PRINT '
      "APRES LA FORMULE ECRIVE Z GO TO 1"
1030  RETURN
1100  FOR J= 1 TO 1500: NEXT J
1110  RETURN
2000  IF W >= 1 GOTO 2050
2010  FOR J= 1 TO 200
2020  XX= PEEK (- 16336):NEXT J
2030  HOME: VTAB 5: PRINT  VOUS N'AVEZ PAS PREPAREZ
2035  VTAB 10: PRINT TAB (10) " AU REVOIR "
2040  END
2050  HOME: VTAB 10: PRINT TAB (6):
      VOUS AVEZ REÇUS ; W; DE KI

```

خاتمة

هذا هو أهم ما جاء بلغة « بازيك » ، وضعبته باختصار شديد مع كثير من الأمثلة . ويجدر الإشارة إلى وجود بعض الفروقات في اللغة حسب الآلة وحسب نظام التشغيل أو المفسرة المتبعة في الشركة المصنعة ، ويجب على كل مبرمج أن يطلع عليها من الوثائق التابعة للآلة والخاصة بهذه اللغة . وخاصة لجهة أوامر الادخال والإخراج والعمل بالأجهزة الخارجية . يبقى أن أشير إلى أن هذه اللغة هي من أكثر اللغات استعمالاً إضافة للغة « باسكال » في الميكروكومبيوتر ، كما وهي من أكثر اللغات شيوعاً في الجامعات لجهة تدريس الطلاب وتدريبهم على البرمجة . وخاصة في برمجة المسائل الرياضية والعلمية .

أما من ناحية البرمجة فلا بد من الإشارة إلى زيادة أهميتها ، خاصة بعد تطور الآلات الحاسبة ، وزهد ثمنها بالنسبة للبرامج المباعة معها (SOFTWARE) . ومن المعلوم أن أهمية الميكروبروسيسور تكمن في برمجته ، ويقدر ما نستطيع من وضع البرنامج الصحيح لعملية تكنولوجية أو إدارية معينة بقدر ما نكون قد استطعنا من إستعماله والإفادة منه بشكل كبير وهنا يكمن وزن وأهمية الميكروبروسيسور والآلات الحاسبة .

يبقى أن أذكر كل من يدرس ويرغب في تعلم البرمجة ، بأن ذلك ليس بالأمر البسيط والسهل . وانه من غير المعقول وضع برنامج بدون أخطاء ومن المرة الأولى . وأن البرمجة بلغة معينة ستصبح سهلة وبسيطة حينما يتم ذلك بشكل مباشر على الآلة ، التي تشير إلى الأخطاء فيتم تصحيحها وفهم سببها .

كما ويجب على المبرمج أن يكون على اطلاع ودراية بموضوع المسائل المطلوب حلها وبرمجتها ، فليس من المعقول وضع برنامج لمسألة كهربائية أو اقتصادية أو في علم الفضاء مثلاً ، دون الإطلاع والإلمام بهذه العلوم .

الفهرس

الموضوع	الصفحة
المقدمة	5
الفصل الأول : البرمجة	9
الفصل الثاني : المدخل الى البرمجة بلغة بازيك	17
الفصل الثالث : الأوامر المستعملة في تنظيم وإدارة الأجهزة والسجلات	81
الفصل الرابع : مسائل برمجة بلغة بازيك	103
خاتمة	157

هذا الكتاب

هذا الكتاب يعطي صورة واضحة عن طريقة البرمجة بلغة « بازيك » (BASIC) . وهو مفيد لطلاب علوم الكمبيوتر والبرمجة ، وللاختصاصيين في هذه الحقول . كما ولكل من يرغب في تعلم البرمجة بهذه اللغة والعمل بها . ويحتوي هذا الكتاب على ثلاثة فصول :

الفصل الأول : عبارة عن مدخل الى البرمجة ، ويعطي صورة موجزة عن البرمجة وكيفية وضع البرامج .

الفصل الثاني : يُعتبر مدخل الى لغة « بازيك » ، ويعطي صورة واضحة عن هذه اللغة مع الكثير من الأمثلة التي تساعد القارئ على فهم هذه اللغة والعمل بها .

الفصل الثالث : ويحتوي على الأوامر والتعليمات المتعلقة بتنظيم وإدارة الأجهزة والسجلات .

الفصل الرابع : ويحتوي على عدد كبير من المسائل المحلولة والمبرمجة ، التي تساعد القارئ في عمله بهذه اللغة .

ولقد حاولت قدر الإمكان الحفاظ على المصطلحات الإنكليزية المعتمدة في الكمبيوتر ، بالإضافة إلى المصطلحات العربية الجديدة التي دخلت الى هذا العلم منذ وقت قصير .